using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity.Validation;

using KM.JXC.DBA;

using KM.JXC.Common.KMException;

using KM.JXC.BL.Open.Interface;

using KM.JXC.BL.Open.TaoBao;

using KM.JXC.Common.Util;

using KM.JXC.BL.Models;

namespace KM.JXC.BL

{

public class AccessManager

{

public IAccessToken TokenManager{get;set;}

public IOShopManager ShopManager { get; private set; }

public IOUserManager MallUserManager { get; private set; }

public UserManager LocUserManager { get; private set; }

public int Mall\_Type\_ID { get; set; }

public AccessManager(int mall\_type\_id)

{

this.Mall\_Type\_ID = mall\_type\_id;

TokenManager = new TaoBaoAccessToken(this.Mall\_Type\_ID);

}

/// <summary>

/// Initialize IManager instances

/// </summary>

/// <param name="token">Access token got from IAccessToken</param>

private void InitializeMallManagers(Access\_Token token)

{

this.ShopManager = new TaoBaoShopManager(token, this.Mall\_Type\_ID);

this.MallUserManager = new TaoBaoUserManager(token,this.Mall\_Type\_ID);

}

/// <summary>

/// Calback from Mall Open API Authorization, it will verify if current login user has access to the system

/// </summary>

/// <param name="code">returns by Mall Open API Authorization</param>

/// <returns></returns>

public Access\_Token AuthorizationCallBack(string code)

{

Access\_Token request\_token = null;

BUser requester = new BUser();

//must get access token after mall authorization to identify user

request\_token = TokenManager.RequestAccessToken(code);

if (request\_token == null)

{

throw new KMJXCException("没有获取到Access token", ExceptionLevel.SYSTEM);

}

requester.Type = new BMallType() { ID = this.Mall\_Type\_ID };

requester.Mall\_ID = request\_token.Mall\_User\_ID;

requester.Mall\_Name = request\_token.Mall\_User\_Name;

requester.Parent\_ID = 0;

requester.Parent = null;

KuanMaiEntities db = new KuanMaiEntities();

try

{

var db\_user = from u in db.User

where u.Mall\_ID == requester.Mall\_ID && u.Mall\_Name == requester.Mall\_Name && u.Mall\_Type == this.Mall\_Type\_ID

select new BUser

{

ID = u.User\_ID,

Name = u.Name,

Mall\_Name = u.Mall\_Name,

Mall\_ID = u.Mall\_ID,

Password = u.Password,

Parent\_ID = (int)u.Parent\_User\_ID,

};

List<BUser> users = db\_user.ToList<BUser>();

//Create user in local db with mall owner id

if (users.Count == 0)

{

this.InitializeMallManagers(request\_token);

if (this.ShopManager == null)

{

throw new KMJXCException("IShopManager 实例为null", ExceptionLevel.SYSTEM);

}

//check if current user's shop is ready in system

Shop shop = this.ShopManager.GetShop(requester);

if (shop == null)

{

BUser subUser = this.MallUserManager.GetSubUser(requester.Mall\_ID, requester.Mall\_Name);

if (subUser == null)

{

throw new KMJXCException("用户:" + requester.Mall\_Name + " 没有对应的" + ((KM.JXC.BL.Open.OBaseManager)this.ShopManager).MallType.Description + ",并且不属于任何店铺的子账户", ExceptionLevel.ERROR);

}

else

{

if (subUser.Parent == null || string.IsNullOrEmpty(subUser.Parent.Mall\_Name))

{

throw new KMJXCException("用户:" + requester.Mall\_Name + " 没有对应的" + ((KM.JXC.BL.Open.OBaseManager)this.ShopManager).MallType.Description + ",并且不属于任何店铺的子账户", ExceptionLevel.ERROR);

}

BUser mainUser = null;

var u = from us in db.User

where us.Mall\_ID == subUser.Parent.Mall\_ID && us.Mall\_Type == requester.Type.ID && us.Mall\_Name == subUser.Parent.Mall\_Name

select new BUser

{

ID = us.User\_ID,

Name = us.Name,

Mall\_Name = us.Mall\_Name,

Mall\_ID = us.Mall\_ID,

Password = us.Password,

Parent\_ID = (int)us.Parent\_User\_ID,

Type = new BMallType { ID = us.Mall\_Type }

};

if (u.ToList<BUser>().Count() == 1)

{

mainUser = u.ToList<BUser>()[0];

}

if (mainUser == null)

{

throw new KMJXCException("主账户:" + subUser.Parent.Mall\_Name + " 还没有初始化店铺信息，所有子账户无法登录系统", ExceptionLevel.ERROR);

}

requester.Parent\_ID = mainUser.ID;

requester.Parent = mainUser;

requester.EmployeeInfo = subUser.EmployeeInfo;

}

}

//create user in local db

requester.Name = requester.Mall\_Name;

requester.Password = Guid.NewGuid().ToString();

User dbUser = new User();

dbUser.User\_ID = requester.ID;

dbUser.Mall\_ID = requester.Mall\_ID;

dbUser.Mall\_Name = requester.Mall\_Name;

dbUser.Name = requester.Name;

dbUser.Mall\_Type = requester.Type.ID;

dbUser.Parent\_Mall\_ID = "";

dbUser.Parent\_Mall\_Name = "";

dbUser.Parent\_User\_ID = 0;

dbUser.Password = "";

dbUser.Name = dbUser.Mall\_Name;

dbUser.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

dbUser.Modified = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

dbUser.Modified\_By = 0;

if (requester.Parent != null)

{

dbUser.Parent\_Mall\_ID = requester.Parent.Mall\_ID;

dbUser.Parent\_Mall\_Name = requester.Parent.Mall\_Name;

dbUser.Parent\_User\_ID = requester.Parent.ID;

}

db.User.Add(dbUser);

db.SaveChanges();

//create access token for the new user

request\_token.User\_ID = dbUser.User\_ID;

requester.ID = dbUser.User\_ID;

db.Access\_Token.Add(request\_token);

//save employee

if (requester.Parent\_ID > 0 && requester.EmployeeInfo != null)

{

requester.EmployeeInfo.User\_ID = requester.ID;

Employee employee = new Employee();

employee.Name = requester.EmployeeInfo.Name;

employee.IdentityCard = requester.EmployeeInfo.IdentityCard;

employee.MatureDate = requester.EmployeeInfo.MatureDate;

employee.Phone = requester.EmployeeInfo.Phone;

employee.User\_ID=requester.EmployeeInfo.User\_ID;

employee.HireDate = requester.EmployeeInfo.HireDate;

employee.Gendar = requester.EmployeeInfo.Gendar;

employee.Duty = requester.EmployeeInfo.Duty;

employee.Email = requester.EmployeeInfo.Email;

employee.Department = requester.EmployeeInfo.Department;

employee.BirthDate = requester.EmployeeInfo.BirthDate;

employee.Address = requester.EmployeeInfo.Address;

db.Employee.Add(employee);

}

if (shop != null)

{

//create local shop information for the new main user

shop.User\_ID = requester.ID;

shop.Parent\_Shop\_ID = 0;

shop.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

shop.Synced = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

db.Shop.Add(shop);

db.SaveChanges();

//save shop user

Shop\_User shop\_User = new Shop\_User();

shop\_User.User\_ID = requester.ID;

shop\_User.Shop\_ID = shop.Shop\_ID;

db.Shop\_User.Add(shop\_User);

//update dbuser

dbUser.Shop\_ID = shop.Shop\_ID;

db.SaveChanges();

//create default stock house

Store\_House shouse = new Store\_House();

shouse.Shop\_ID = shop.Shop\_ID;

shouse.Title = "默认仓库";

shouse.User\_ID = requester.ID;

shouse.Create\_Time = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

db.Store\_House.Add(shouse);

db.SaveChanges();

}

}

else

{

//Verify if local db has non expried accesstoken

requester = users[0];

Access\_Token local\_token = GetLocalToken(requester.ID, this.Mall\_Type\_ID);

if (local\_token != null)

{

long timeNow = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now); //last access token is expried

if (timeNow >= local\_token.Expirse\_In + local\_token.Request\_Time)

{

request\_token = TokenManager.RequestAccessToken(code);

request\_token.User\_ID = requester.ID;

UpdateLocalAccessToken(request\_token);

}

else

{

request\_token = local\_token;

}

}

}

}

catch (DbEntityValidationException dbex)

{ }

catch (Exception ex)

{

throw new KMJXCException(ex.Message, ExceptionLevel.SYSTEM);

}

finally

{

if (db != null)

{

db.Dispose();

}

}

return request\_token;

}

public Access\_Token GetAccessToken(BUser loginUser)

{

Access\_Token token = null;

token = this.GetLocalToken(loginUser.ID, this.Mall\_Type\_ID);

if (token != null)

{

long timeNow = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now); //last access token is expried

if (timeNow >= token.Expirse\_In + token.Request\_Time)

{

return null;

}

else

{

return token;

}

} return null;

}

/// Get access token from local db

/// </summary>

/// <param name="user\_id"></param>

/// <param name="mall\_type\_id"></param>

/// <returns></returns>

private Access\_Token GetLocalToken(long user\_id, int mall\_type\_id)

{

Access\_Token token = null; KuanMaiEntities db = new KuanMaiEntities(); var etoken = from p in db.Access\_Token where p.Mall\_Type\_ID == mall\_type\_id && p.User\_ID == user\_id select p; if (etoken != null)

{

token = etoken.ToList<Access\_Token>()[0];

} return token;

} /// <summary>

/// Update access token in local db

/// </summary>

/// <param name="token"></param>

/// <returns></returns>

private bool UpdateLocalAccessToken(Access\_Token token)

{

bool result = false;

Access\_Token local\_token = null;

KuanMaiEntities db = new KuanMaiEntities();

var etoken = from p in db.Access\_Token where p.Mall\_Type\_ID == token.Mall\_Type\_ID && p.User\_ID == token.User\_ID select p;

if (etoken != null)

{

local\_token = etoken.ToList<Access\_Token>()[0];

}

if (local\_token != null)

{

local\_token.Access\_Token1 = token.Access\_Token1;

local\_token.Expirse\_In = token.Expirse\_In;

local\_token.Request\_Time = token.Request\_Time;

local\_token.RExpirse\_In = token.RExpirse\_In;

db.Access\_Token.Attach(local\_token);

db.SaveChanges();

result = true;

}

return result;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity;

using System.Data;using KM.JXC.DBA;

using KM.JXC.Common.KMException;

using KM.JXC.Common.Util;

using KM.JXC.BL.Open.Interface;

using KM.JXC.BL.Open.TaoBao;

using KM.JXC.BL.Models;namespace KM.JXC.BL

{

public class SalesManager : BBaseManager

{

private StockManager stockManager = null;

private IOTradeManager tradeManager = null;

public SalesManager(BUser user, int shop\_id, Permission permission)

: base(user, shop\_id, permission)

{

tradeManager = new TaobaoTradeManager(this.AccessToken, this.Shop.Mall\_Type\_ID);

} public SalesManager(BUser user, Shop shop, Permission permission)

: base(user, shop, permission)

{

stockManager = new StockManager(user, shop, permission);

tradeManager = new TaobaoTradeManager(this.AccessToken, this.Shop.Mall\_Type\_ID);

}

public BSyncTime GetSyncTime(int syncType=0)

{

BSyncTime time = null;

using (KuanMaiEntities db = new KuanMaiEntities())

{

time = (from stime in db.Sale\_SyncTime

join user in db.User on stime.SyncUser equals user.User\_ID into luser

from l\_user in luser.DefaultIfEmpty()

where stime.ShopID == this.Shop.Shop\_ID && stime.SyncType == syncType

orderby stime.LastSyncTime descending

select new BSyncTime

{

Last = stime.LastSyncTime,

LastTradeModifiedEndTime=stime.LastTradeModifiedEndTime,

LastTradeStartEndTime=stime.LastTradeStartEndTime,

SyncUser = new BUser

{

ID = l\_user.User\_ID,

Name = l\_user.Name,

Mall\_ID = l\_user.Mall\_ID,

Mall\_Name = l\_user.Mall\_Name

}

}).FirstOrDefault<BSyncTime>();

}

return time;

}

/// <summary>

/// The back sales won't be created while the sales don't has corresponding leave stocks

/// Example:Customers refounded before expressed

/// </summary>

/// <param name="trades"></param>

private void HandleBackTrades(List<BSale> trades, Shop shop)

{

if (trades == null || trades.Count == 0 || shop==null)

{

return;

}

KuanMaiEntities db = new KuanMaiEntities();

try

{

string[] sale\_ids=(from trade in trades select trade.Sale\_ID).ToArray<string>();

List<Leave\_Stock> leave\_Stocks=(from ls in db.Leave\_Stock where sale\_ids.Contains(ls.Sale\_ID) select ls).ToList<Leave\_Stock>();

int[] ls\_ids=(from ls in leave\_Stocks select ls.Leave\_Stock\_ID).ToArray<int>();

List<Leave\_Stock\_Detail> leave\_stock\_Details=(from lsd in db.Leave\_Stock\_Detail where ls\_ids.Contains(lsd.Leave\_Stock\_ID) select lsd).ToList<Leave\_Stock\_Detail>();

List<Back\_Sale> back\_Sales=(from backSale in db.Back\_Sale where sale\_ids.Contains(backSale.Sale\_ID) select backSale).ToList<Back\_Sale>();

int[] bs\_ids=(from bs in back\_Sales select bs.Back\_Sale\_ID).ToArray<int>();

List<Back\_Sale\_Detail> back\_Sale\_Details=(from bsd in db.Back\_Sale\_Detail where bs\_ids.Contains(bsd.Back\_Sale\_ID) select bsd).ToList<Back\_Sale\_Detail>();

List<Sale\_Detail> saleDetails=(from sd in db.Sale\_Detail where sale\_ids.Contains(sd.Mall\_Trade\_ID) select sd).ToList<Sale\_Detail>();

foreach (BSale trade in trades)

{

Leave\_Stock ls=(from leaveStock in leave\_Stocks where leaveStock.Sale\_ID==trade.Sale\_ID select leaveStock).FirstOrDefault<Leave\_Stock>();

//no leave stock no need to create back sale

if (ls == null)

{

continue;

}

Back\_Sale bSale=(from b\_Sale in back\_Sales where b\_Sale.Sale\_ID==trade.Sale\_ID select b\_Sale).FirstOrDefault<Back\_Sale>();

bool isNewBackSale = false;

double totalRefound = 0;

if (bSale == null)

{

isNewBackSale = true;

bSale = new Back\_Sale();

bSale.Back\_Date = trade.Synced;

bSale.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

bSale.Description = "";

bSale.Sale\_ID = trade.Sale\_ID;

bSale.Shop\_ID = shop.Shop\_ID;

bSale.Status = 0;

bSale.User\_ID = this.CurrentUser.ID;

}

if (isNewBackSale)

{

db.Back\_Sale.Add(bSale);

db.SaveChanges();

}

if (trade.Orders != null)

{

foreach (BOrder order in trade.Orders)

{

Sale\_Detail sale\_detail=(from odetail in saleDetails where odetail.Mall\_Order\_ID==order.Order\_ID select odetail).FirstOrDefault<Sale\_Detail>();

if (sale\_detail == null)

{

continue;

}

//1- refound succeed

//0- is normal

if (!order.Refound)

{

continue;

}

Back\_Sale\_Detail dbSaleDetail = (from bsd in back\_Sale\_Details where bsd.Order\_ID==order.Order\_ID select bsd).FirstOrDefault<Back\_Sale\_Detail>();

Leave\_Stock\_Detail dbLeaveStockDetail=(from dblsd in leave\_stock\_Details where dblsd.Order\_ID==order.Order\_ID select dblsd).FirstOrDefault<Leave\_Stock\_Detail>();

//no need to create back sale detail while no leave stock detail

if (dbSaleDetail == null && dbLeaveStockDetail!=null)

{

dbSaleDetail = new Back\_Sale\_Detail();

dbSaleDetail.Order\_ID = order.Order\_ID;

dbSaleDetail.Back\_Sale\_ID = bSale.Back\_Sale\_ID;

dbSaleDetail.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

dbSaleDetail.Description = "同步商城订单时处理有退货的订单(有出库单详细信息)";

dbSaleDetail.Parent\_Product\_ID = dbLeaveStockDetail.Parent\_Product\_ID;

dbSaleDetail.Price = order.Price;

dbSaleDetail.Product\_ID = dbLeaveStockDetail.Product\_ID;

dbSaleDetail.Quantity = order.Quantity;

dbSaleDetail.Status = 0;

dbSaleDetail.Refound = order.Amount;

totalRefound += order.Amount;

db.Back\_Sale\_Detail.Add(dbSaleDetail);

sale\_detail.Status1 = (int)SaleDetailStatus.REFOUNDED\_WAIT\_HANDLE;

}

}

bSale.Amount = totalRefound;

}

}

base.CreateActionLog(new BUserActionLog() { Shop=new BShop{ ID=shop.Shop\_ID}, Action = new BUserAction() { Action\_ID = UserLogAction.CREATE\_BACK\_SALE }, Description = "同步商城订单到进销存，成功调用商城API,有退货交易，进销存退货单成功创建" });

db.SaveChanges();

}

catch (Exception ex)

{

}

finally

{

db.Dispose();

}

}

/// <summary>

/// The leave stocks won't be created while orders products are not mapped with local products

/// </summary>

/// <param name="trades"></param>

private void CreateLeaveStocks(List<BSale> trades,Shop shop)

{

if (trades == null)

{

return;

}

if (shop == null)

{

return;

}

KuanMaiEntities db = new KuanMaiEntities();

int[] csp\_ids = (from child in this.DBChildShops select child.Shop\_ID).ToArray<int>();

if (csp\_ids == null)

{

csp\_ids = new int[1];

}

List<Product> allproducts = (from pdt in db.Product where (pdt.Shop\_ID == shop.Shop\_ID || pdt.Shop\_ID == this.Main\_Shop.Shop\_ID || csp\_ids.Contains(pdt.Shop\_ID)) select pdt).ToList<Product>();

string[] sale\_ids=(from sale in trades select sale.Sale\_ID).ToArray<string>();

List<Leave\_Stock> cacheLeaveStocks=(from ls in db.Leave\_Stock where sale\_ids.Contains(ls.Sale\_ID) select ls).ToList<Leave\_Stock>();

Store\_House house = (from store in db.Store\_House where store.Default == true && (store.Shop\_ID == shop.Shop\_ID || store.Shop\_ID == this.Main\_Shop.Shop\_ID || csp\_ids.Contains(store.Shop\_ID)) select store).FirstOrDefault<Store\_House>();

List<Store\_House> houses = (from store in db.Store\_House select store).ToList<Store\_House>();

List<Stock\_Pile> stockPiles = (from sp in db.Stock\_Pile where sp.Shop\_ID == shop.Shop\_ID || sp.Shop\_ID == this.Main\_Shop.Shop\_ID || csp\_ids.Contains(sp.Shop\_ID) select sp).ToList<Stock\_Pile>();

List<Sale\_Detail> tradeDetails=(from tradeDetail in db.Sale\_Detail where sale\_ids.Contains(tradeDetail.Mall\_Trade\_ID) select tradeDetail).ToList<Sale\_Detail>();

List<Stock\_Batch> batches=(from b in db.Stock\_Batch where b.ShopID==shop.Shop\_ID select b).ToList<Stock\_Batch>();

try

{

foreach (BSale trade in trades)

{

bool isNew = false;

Leave\_Stock dbStock = null;

dbStock=(from cstock in cacheLeaveStocks where cstock.Sale\_ID==trade.Sale\_ID select cstock).FirstOrDefault<Leave\_Stock>();

if (dbStock == null)

{

isNew = true;

dbStock = new Leave\_Stock();

dbStock.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

dbStock.Leave\_Date = trade.Synced;

dbStock.Sale\_ID = trade.Sale\_ID;

dbStock.Shop\_ID = shop.Shop\_ID;

dbStock.User\_ID = this.CurrentUser.ID;

}

if (trade.Orders != null)

{

List<Leave\_Stock\_Detail> dbLDetails = new List<Leave\_Stock\_Detail>();

#region handle sale orders

foreach (BOrder order in trade.Orders)

{

Leave\_Stock\_Detail dbDetail = new Leave\_Stock\_Detail();

int stockPileProductId = 0;

Sale\_Detail order\_detail = (from orderDetail in tradeDetails

where orderDetail.Mall\_Trade\_ID == trade.Sale\_ID && orderDetail.Mall\_Order\_ID == order.Order\_ID

select orderDetail).FirstOrDefault<Sale\_Detail>();

if (order\_detail == null)

{

continue;

}

order\_detail.SyncResultMessage = "";

if (order.Refound)

{

order\_detail.Status1 = (int)SaleDetailStatus.REFOUND\_BEFORE\_SEND;

order\_detail.SyncResultMessage = "已经退货，不需要出库";

db.SaveChanges();

continue;

}

Product parentPdt = (from pdt in allproducts where pdt.Product\_ID == order.Parent\_Product\_ID

select pdt).FirstOrDefault<Product>();

Product childPdt = (from pdt in allproducts where pdt.Product\_ID == order.Product\_ID

select pdt).FirstOrDefault<Product>();

order.Parent\_Product\_ID = 0;

if (parentPdt != null)

{

order.Parent\_Product\_ID = parentPdt.Product\_ID;

if (childPdt != null)

{

order.Product\_ID = childPdt.Product\_ID;

}

else

{

if (!string.IsNullOrEmpty(order.Mall\_SkuID))

{

order\_detail.Status1 = (int)SaleDetailStatus.NOT\_CONNECTED;

order\_detail.SyncResultMessage = "宝贝已关联，当前SKU未关联，不能出库";

db.SaveChanges();

continue;

}

}

}

else

{

if (childPdt != null)

{

order.Parent\_Product\_ID = childPdt.Parent\_ID;

order.Product\_ID = childPdt.Product\_ID;

}

}

//no need to create leave stock while the mall product is not mapped with local product

if (order.Product\_ID == 0 && order.Parent\_Product\_ID == 0)

{

order\_detail.Status1 = (int)SaleDetailStatus.NOT\_CONNECTED;

order\_detail.SyncResultMessage = "宝贝未关联，不能更新库存";

db.SaveChanges();

continue;

}

stockPileProductId = order.Product\_ID;

if (stockPileProductId == 0 && string.IsNullOrEmpty(order.Mall\_SkuID) && order.Parent\_Product\_ID>0)

{

stockPileProductId = order.Parent\_Product\_ID;

}

dbDetail.Leave\_Stock\_ID = dbStock.Leave\_Stock\_ID;

dbDetail.Price = order.Price;

dbDetail.Quantity = order.Quantity;

dbDetail.Amount = order.Amount;

Stock\_Pile stockPile = null;

if (house != null)

{

//order\_detail.SyncResultMessage = "默认仓库：" + house.Title;

//create leave stock from default store house

stockPile = (from sp in stockPiles

where sp.Product\_ID == stockPileProductId && sp.StockHouse\_ID == house.StoreHouse\_ID && sp.Quantity >= order.Quantity

select sp).OrderBy(s=>s.Batch\_ID).FirstOrDefault<Stock\_Pile>();

}

if (stockPile == null)

{

//get store house when it has the specific product

var tmpstockPile = from sp in stockPiles

where sp.Product\_ID == stockPileProductId && sp.Quantity >= order.Quantity select sp;

if (tmpstockPile.Count() > 0)

{

stockPile = tmpstockPile.OrderBy(s=>s.Batch\_ID).ToList<Stock\_Pile>()[0];

Store\_House tmpHouse = (from h in houses where h.StoreHouse\_ID == stockPile.StockHouse\_ID

select h).FirstOrDefault<Store\_House>();

if (tmpHouse != null)

{

house = tmpHouse;

}

}

else

{

//cannot leave stock, no stock pile

order\_detail.Status1 = (int)SaleDetailStatus.NO\_ENOUGH\_STOCK;

order\_detail.SyncResultMessage = "没有足够的库存，不能出库";

}

}

//no stock cannot leave stock

if (stockPile != null)

{

order\_detail.Status1 = (int)SaleDetailStatus.LEAVED\_STOCK;

order\_detail.SyncResultMessage = "出库仓库：" + house.Title;

dbDetail.Parent\_Product\_ID = order.Parent\_Product\_ID;

dbDetail.Product\_ID = order.Product\_ID;

dbDetail.StoreHouse\_ID = stockPile.StockHouse\_ID;

dbDetail.Batch\_ID = stockPile.Batch\_ID;

//Update stock

stockPile.Quantity = stockPile.Quantity - order.Quantity;

//Update stock field in Product table

Product product=(from pdt in allproducts where pdt.Product\_ID==dbDetail.Parent\_Product\_ID

select pdt).FirstOrDefault<Product>();

if (product != null)

{

product.Quantity = product.Quantity - order.Quantity;

}

dbDetail.Order\_ID = order.Order\_ID;

dbLDetails.Add(dbDetail);

//db.Leave\_Stock\_Detail.Add(dbDetail);

}

}

#endregion

if (isNew && dbLDetails.Count>0)

{

db.Leave\_Stock.Add(dbStock);

db.SaveChanges();

foreach (Leave\_Stock\_Detail d in dbLDetails)

{

if (d.Leave\_Stock\_ID == 0)

{

d.Leave\_Stock\_ID = dbStock.Leave\_Stock\_ID;

}

db.Leave\_Stock\_Detail.Add(d);

}

}

}

}

base.CreateActionLog(new BUserActionLog() { Shop=new BShop{ ID=shop.Shop\_ID}, Action = new BUserAction() { Action\_ID = UserLogAction.CREATE\_LEAVE\_STOCK }, Description = "同步商城订单到进销存，成功调用商城API,未出库的订单已成功出库并更新了产品库存" });

db.SaveChanges();

}

catch (KMJXCException kex)

{

throw kex;

}

catch (Exception ex)

{

throw ex;

}

finally

{

db.Dispose();

}

}

public void SyncMallTrades(int startTime, int endTime, string status, int syncType, int shop\_id = 0)

{

if (shop\_id == 0)

{

this.SyncMallTrades(startTime, endTime, status, syncType, this.Shop);

foreach (Shop s in this.DBChildShops)

{

this.SyncMallTrades(startTime, endTime, status, syncType, s);

}

}

else

{

Shop tmpShop = null;

int[] child\_shop\_ids=(from c in this.DBChildShops select c.Shop\_ID).ToArray<int>();

if (this.Shop.Shop\_ID != shop\_id && !child\_shop\_ids.Contains(shop\_id))

{

throw new KMJXCException("不能同步他人店铺的订单，不要再次尝试此操作");

}

if (this.Main\_Shop.Shop\_ID == shop\_id && this.CurrentUser.Shop.ID != shop\_id)

{

throw new KMJXCException("没有权限同步主店铺订单");

}

else if (this.Main\_Shop.Shop\_ID == shop\_id && this.CurrentUser.Shop.ID == shop\_id)

{

tmpShop = this.Main\_Shop;

}

else if(this.Main\_Shop.Shop\_ID!=shop\_id)

{

if (shop\_id == this.Shop.Shop\_ID)

{

tmpShop = this.Shop;

}

else

{

tmpShop = (from s in this.DBChildShops where s.Shop\_ID == shop\_id select s).FirstOrDefault<Shop>();

}

}

this.SyncMallTrades(startTime, endTime, status, syncType, tmpShop);

}

}

private bool SyncMallTrades(int startTime, int endTime, string status,int syncType,Shop shop)

{

bool result = false;

try

{

KuanMaiEntities db = new KuanMaiEntities();

IOTradeManager tradeManager = null;

Access\_Token token = this.AccessToken;

Shop desshop = this.Shop;

if (shop != null)

{

desshop = shop;

token = (from t in db.Access\_Token

from u in db.User

where t.User\_ID == u.User\_ID && u.User\_ID == shop.User\_ID

select t).FirstOrDefault<Access\_Token>();

if (token == null)

{

throw new KMJXCException("店铺 "+shop.Name+" 的授权已经过期，请使用此店铺掌柜账号登录一次系统");

} if (IsTokenExpired(token))

{

throw new KMJXCException("店铺 " + shop.Name + " 的授权已经过期，请使用此店铺掌柜账号登录一次系统");

}

} tradeManager = new TaobaoTradeManager(token, desshop.Mall\_Type\_ID);

List<BSale> allSales = new List<BSale>();

Sale\_SyncTime syncTime = null;

syncTime = (from sync in db.Sale\_SyncTime where sync.ShopID == desshop.Shop\_ID && sync.SyncType == syncType orderby sync.LastSyncTime descending select sync).FirstOrDefault<Sale\_SyncTime>();

DateTime tNow = DateTime.Now;

long timeNow = DateTimeUtil.ConvertDateTimeToInt(tNow);

long lastSyncModifiedTime = 0;

lastSyncModifiedTime = DateTimeUtil.ConvertDateTimeToInt(tNow.AddDays(-1));

if (syncTime != null)

{

lastSyncModifiedTime = syncTime.LastTradeModifiedEndTime;

}

syncTime = new Sale\_SyncTime();

syncTime.ShopID = desshop.Shop\_ID;

syncTime.SyncUser = this.CurrentUser.ID;

syncTime.LastSyncTime = timeNow;

syncTime.SyncType = syncType;

db.Sale\_SyncTime.Add(syncTime);

//0 is normal sync trade by created time, 1 is increment sync trade by modified time

if (syncType == 0)

{

syncTime.LastTradeStartEndTime = endTime + 1;

syncTime.LastTradeModifiedEndTime = 0;

DateTime? sDate = null;

DateTime? eDate = null;

if (startTime > 0)

{

sDate = DateTimeUtil.ConvertToDateTime(startTime);

} if (endTime > 0)

{

eDate = DateTimeUtil.ConvertToDateTime(endTime);

} allSales = tradeManager.SyncMallTrades(sDate, eDate, status);

}

else if (syncType == 1)

{

syncTime.LastTradeStartEndTime = 0;

syncTime.LastTradeModifiedEndTime = timeNow + 1;

allSales = tradeManager.IncrementSyncMallTrades(lastSyncModifiedTime, timeNow, status);

}

List<BSale> sales = new List<BSale>();

foreach (BSale s in allSales)

{

List<BSale> existed = (from ss in sales where ss.Sale\_ID == s.Sale\_ID select ss).ToList<BSale>();

if (existed == null || existed.Count == 0)

{

sales.Add(s);

}

}

base.CreateActionLog(new BUserActionLog() { Shop = new BShop { ID = shop.Shop\_ID }, Action = new BUserAction() { Action\_ID = UserLogAction.SYNC\_SHOP\_TRADE }, Description = "同步商城订单到进销存，成功调用商城API," + sales.Count+" 条新的交易同步回进销存" });

this.HandleMallTrades(sales, desshop);

db.SaveChanges();

}

catch (KMJXCTaobaoException mex)

{

throw mex;

}

catch (KMJXCMallException mex)

{

throw mex;

}

catch (KMJXCException ex)

{

throw ex;

}

catch (Exception ex)

{

throw ex;

}

return result;

}

/// <summary>

/// Handle sales synchronized from mall including creating leave stocks and handle refounded trades which already synchronized before.

/// </summary>

/// <param name="allSales"></param>

private void HandleMallTrades(List<BSale> allSales,Shop shop)

{

if (allSales == null)

{

return;

}

List<BSale> newSales = new List<BSale>();

List<BSale> backSales = new List<BSale>();

List<Product> allProducts = null;

KuanMaiEntities db = new KuanMaiEntities();

try

{

int[] child\_shops=(from c in this.DBChildShops select c.Shop\_ID).ToArray<int>();

allProducts = (from p in db.Product

where p.Shop\_ID == shop.Shop\_ID || child\_shops.Contains(p.Shop\_ID) || p.Shop\_ID == this.Main\_Shop.Shop\_ID

select p).ToList<Product>();

var tmpExp = from expsp in db.Express\_Shop

join exp in db.Express on expsp.Express\_ID equals exp.Express\_ID into lExp

from l\_exp in lExp

where expsp.Shop\_ID == shop.Shop\_ID && expsp.IsDefault == 1

select l\_exp;

Express defaultExp = tmpExp.FirstOrDefault<Express>();

List<Express\_Fee> expFees = new List<Express\_Fee>();

if (defaultExp != null)

{

expFees = (from fee in db.Express\_Fee

where fee.Express\_ID == defaultExp.Express\_ID && fee.Shop\_ID == shop.Shop\_ID

select fee).ToList<Express\_Fee>();

}

long timeNow = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

var customers = from customer in db.Customer

where customer.Mall\_Type\_ID == shop.Mall\_Type\_ID

select customer;

List<Customer\_Shop> shop\_customers = (from shop\_cus in db.Customer\_Shop where shop\_cus.Shop\_ID == shop.Shop\_ID

select shop\_cus).ToList<Customer\_Shop>();

string[] sale\_ids=(from sale in allSales select sale.Sale\_ID).ToArray<string>();

List<Sale> dbSales = (from sale in db.Sale where sale\_ids.Contains(sale.Mall\_Trade\_ID) select sale).ToList<Sale>();

List<Common\_District> areas = BBaseManager.Areas;

foreach (BSale sale in allSales)

{

Sale dbSale = new Sale();

dbSale.Amount = sale.Amount;

dbSale.Buyer\_ID = 0;

dbSale.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

dbSale.Express\_Cop = "";

dbSale.Mall\_Trade\_ID = sale.Sale\_ID;

dbSale.Modified = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

dbSale.Post\_Fee = sale.Post\_Fee;

dbSale.Province\_ID = 0;

dbSale.City\_ID = 0;

dbSale.Sync\_User = this.CurrentUser.ID;

dbSale.HasRefound = sale.HasRefound;

if (defaultExp != null)

{

dbSale.Express\_ID = defaultExp.Express\_ID;

dbSale.Express\_Cop = defaultExp.Name;

}

List<Express\_Fee> tmpFees = new List<Express\_Fee>();

if (sale.Buyer != null)

{

if (sale.Buyer.Province != null)

{

sale.Buyer.Province = (from p in areas where p.name == sale.Buyer.Province.name

select p).FirstOrDefault<Common\_District>();

if (sale.Buyer.Province != null)

{

dbSale.Province\_ID = sale.Buyer.Province.id;

tmpFees=(from fee in expFees where fee.Province\_ID == dbSale.Province\_ID select fee).ToList<Express\_Fee>();

}

}

if (sale.Buyer.City != null)

{

sale.Buyer.City = (from p in areas where p.name == sale.Buyer.City.name select p).FirstOrDefault<Common\_District>();

if (sale.Buyer.City != null)

{

dbSale.City\_ID = sale.Buyer.City.id;

tmpFees = (from fee in expFees where fee.City\_ID == dbSale.City\_ID select fee).ToList<Express\_Fee>();

}

}

if (!string.IsNullOrEmpty(sale.Buyer.Mall\_ID))

{

Customer cus = (from custo in customers where custo.Mall\_ID == sale.Buyer.Mall\_ID

select custo).FirstOrDefault<Customer>();

if (cus == null)

{

cus = new Customer();

cus.Address = sale.Buyer.Address;

cus.Name = sale.Buyer.Name;

cus.City\_ID = dbSale.City\_ID;

cus.Province\_ID = dbSale.Province\_ID;

cus.Mall\_ID = sale.Buyer.Mall\_ID;

cus.Mall\_Type\_ID = this.Shop.Mall\_Type\_ID;

cus.Phone = sale.Buyer.Phone;

db.Customer.Add(cus);

db.SaveChanges();

Customer\_Shop cs = new Customer\_Shop() { Shop\_ID = shop.Shop\_ID, Customer\_ID = cus.Customer\_ID };

db.Customer\_Shop.Add(cs);

}

else

{

//update customer info

cus.Address = sale.Buyer.Address;

cus.Name = sale.Buyer.Name;

cus.City\_ID = dbSale.City\_ID;

cus.Province\_ID = dbSale.Province\_ID;

cus.Mall\_ID = sale.Buyer.Mall\_ID;

cus.Mall\_Type\_ID = this.Shop.Mall\_Type\_ID;

cus.Phone = sale.Buyer.Phone;

Customer\_Shop cuss = (from cusshop in shop\_customers

where cusshop.Customer\_ID == cus.Customer\_ID && cusshop.Shop\_ID == shop.Shop\_ID select cusshop).FirstOrDefault<Customer\_Shop>();

if (cuss == null)

{

Customer\_Shop cs = new Customer\_Shop() { Shop\_ID = shop.Shop\_ID, Customer\_ID = cus.Customer\_ID };

db.Customer\_Shop.Add(cs);

}

}

dbSale.Buyer\_ID = cus.Customer\_ID;

}

}

if (tmpFees != null && tmpFees.Count > 0)

{

dbSale.Post\_Fee1 = tmpFees[0].Fee;

}

dbSale.Sale\_Time = sale.SaleDateTime;

dbSale.Shop\_ID = shop.Shop\_ID;

dbSale.Status = sale.Status;

dbSale.StockStatus = 0;

dbSale.Synced = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

Sale existed = null;

if (dbSales != null && dbSales.Count>0)

{

existed = (from s in dbSales where s.Mall\_Trade\_ID == dbSale.Mall\_Trade\_ID select s).FirstOrDefault<Sale>();

}

if (sale.HasRefound)

{

backSales.Add(sale);

}

if (existed == null)

{

newSales.Add(sale);

db.Sale.Add(dbSale);

if (sale.Orders != null)

{

foreach (BOrder order in sale.Orders)

{

Sale\_Detail sd = new Sale\_Detail();

sd.Amount = order.Amount;

sd.Discount = order.Discount;

sd.Mall\_Order\_ID = order.Order\_ID;

sd.Mall\_Trade\_ID = sale.Sale\_ID;

sd.Refound = order.Refound;

sd.Parent\_Product\_ID = 0;

sd.Product\_ID = 0;

sd.Title = order.Title;

if (string.IsNullOrEmpty(sd.Title))

{

sd.Title = "";

}

Product parentPdt = (from pdt in allProducts where pdt.Product\_ID == order.Parent\_Product\_ID

select pdt).FirstOrDefault<Product>();

Product childPdt = (from pdt in allProducts where pdt.Product\_ID == order.Product\_ID

select pdt).FirstOrDefault<Product>();

if (parentPdt != null)

{

sd.Parent\_Product\_ID = parentPdt.Product\_ID;

}

else

{

if (childPdt != null)

{

sd.Parent\_Product\_ID = childPdt.Parent\_ID;

}

}

if (childPdt != null)

{

sd.Product\_ID = childPdt.Product\_ID;

}

sd.Price = order.Price;

sd.Quantity = order.Quantity;

sd.Status = order.Status;

if (string.IsNullOrEmpty(sd.Status))

{

sd.Status = "0";

}

sd.ImageUrl = order.ImageUrl;

sd.Status1 = order.Status1;

sd.StockStatus = 0;

sd.Mall\_PID = order.Mall\_PID;

sd.Mall\_SkuID = order.Mall\_SkuID;

sd.Supplier\_ID = 0;

sd.Refound = order.Refound;

db.Sale\_Detail.Add(sd);

}

}

}

else

{

List<Sale\_Detail> details = (from detail in db.Sale\_Detail

where detail.Mall\_Trade\_ID == dbSale.Mall\_Trade\_ID

select detail).ToList<Sale\_Detail>();

existed.Modified = dbSale.Modified;

existed.Status = dbSale.Status;

foreach (BOrder order in sale.Orders)

{

Sale\_Detail sd = (from ed in details where ed.Mall\_Order\_ID == order.Order\_ID select ed).FirstOrDefault<Sale\_Detail>();

if (sd != null)

{

continue;

}

sd = new Sale\_Detail();

sd.Amount = order.Amount;

sd.Discount = order.Discount;

sd.Mall\_Order\_ID = order.Order\_ID;

sd.Mall\_Trade\_ID = sale.Sale\_ID;

Product parentPdt = (from pdt in allProducts where pdt.Product\_ID == order.Parent\_Product\_ID

select pdt).FirstOrDefault<Product>();

Product childPdt = (from pdt in allProducts where pdt.Product\_ID == order.Product\_ID

select pdt).FirstOrDefault<Product>();

if (parentPdt != null)

{

sd.Parent\_Product\_ID = parentPdt.Product\_ID;

}

else

{

if (childPdt != null)

{

sd.Parent\_Product\_ID = childPdt.Parent\_ID;

}

}

if (childPdt != null)

{

sd.Product\_ID = childPdt.Product\_ID;

}

sd.Price = order.Price;

sd.Quantity = order.Quantity;

sd.Status = order.Status;

if (string.IsNullOrEmpty(sd.Status))

{

sd.Status = "0";

}

sd.ImageUrl = order.ImageUrl;

//sd.Status1 = order.Status1;

sd.StockStatus = 0;

sd.Mall\_PID = order.Mall\_PID;

sd.Supplier\_ID = 0;

sd.Refound = order.Refound;

db.Sale\_Detail.Add(sd);

}

}

}

db.SaveChanges();

this.CreateLeaveStocks(newSales, shop);

this.HandleBackTrades(backSales,shop);

}

catch (Exception ex)

{

}

finally

{

db.Dispose();

}

}

/// <summary>

/// Search trades

/// </summary>

/// <param name="customers"></param>

/// <param name="sSaleTime"></param>

/// <param name="eSaleTime"></param>

/// <param name="page"></param>

/// <param name="pageSize"></param>

/// <param name="totalRecords"></param>

/// <returns></returns>

public List<BSale> SearchSales(int[] pdtids, string productName,string[] trade\_num,string[] trade\_status, int[] customers, string customer\_nick, long sSaleTime, long eSaleTime, int page, int pageSize, out int totalRecords, int shop\_id = 0)

{

List<BSale> sales = null;

totalRecords = 0;

using (KuanMaiEntities db = new KuanMaiEntities())

{

int[] cspids = (from c in this.DBChildShops select c.Shop\_ID).ToArray<int>();

var trades = from sale in db.Sale

select sale; if (shop\_id == 0)

{

if (this.Shop.Shop\_ID == this.Main\_Shop.Shop\_ID)

{

trades = trades.Where(t => (t.Shop\_ID == this.Shop.Shop\_ID || cspids.Contains(t.Shop\_ID)));

}

else

{

trades = trades.Where(t => t.Shop\_ID == this.Shop.Shop\_ID);

}

}

else

{

trades = trades.Where(t => t.Shop\_ID ==shop\_id);

} if (pdtids != null && pdtids.Length > 0)

{

string[] sale\_ids = (from order in db.Sale\_Detail

where pdtids.Contains(order.Parent\_Product\_ID)

select order.Mall\_Trade\_ID).Distinct<string>().ToArray<string>();

if (sale\_ids != null && sale\_ids.Length > 0)

{

trades = trades.Where(t => sale\_ids.Contains(t.Mall\_Trade\_ID));

}

}

if (!string.IsNullOrEmpty(productName))

{

string[] sale\_ids = (from order in db.Sale\_Detail

join product in db.Product on order.Parent\_Product\_ID equals product.Product\_ID

where product.Parent\_ID==0 && product.Name.Contains(productName)

select order.Mall\_Trade\_ID).Distinct<string>().ToArray<string>();

if (sale\_ids != null && sale\_ids.Length > 0)

{

trades = trades.Where(t => sale\_ids.Contains(t.Mall\_Trade\_ID));

}

} if (trade\_num != null && trade\_num.Length > 0)

{

trades = trades.Where(t =>trade\_num.Contains(t.Mall\_Trade\_ID));

} if (trade\_status != null && trade\_status.Length > 0)

{

trades = trades.Where(t => trade\_status.Contains(t.Status));

} if (customers != null && customers.Length > 0)

{

trades = trades.Where(t => customers.Contains((int)t.Buyer\_ID));

} if(!string.IsNullOrEmpty(customer\_nick))

{

int buyer\_id=(from customer in db.Customer where customer.Mall\_ID==customer\_nick

select customer.Customer\_ID).FirstOrDefault<int>();

trades = trades.Where(t => t.Buyer\_ID==buyer\_id);

} if (sSaleTime > 0)

{

trades = trades.Where(t => t.Sale\_Time >= sSaleTime);

} if (eSaleTime > 0)

{

trades = trades.Where(t => t.Sale\_Time <= eSaleTime);

} var sObjs = from t in trades

join customer in db.Customer on t.Buyer\_ID equals customer.Customer\_ID

join shop in db.Shop on t.Shop\_ID equals shop.Shop\_ID

join user in db.User on t.Sync\_User equals user.User\_ID //into l\_user

//from ll\_user in l\_user.DefaultIfEmpty()

//join province in db.Common\_District on customer.Province\_ID equals province.id

join city in db.Common\_District on customer.City\_ID equals city.id into l\_city

from ll\_city in l\_city.DefaultIfEmpty()

select new BSale

{

Created = (int)t.Created,

Synced = (int)t.Synced,

Modified = (int)t.Modified,

SaleDateTime = (int)t.Sale\_Time,

Amount = t.Amount,

Buyer = new BCustomer

{

ID = customer.Customer\_ID,

Name = customer.Name,

Mall\_ID = customer.Mall\_ID,

Mall\_Name = customer.Mall\_Name,

Phone = customer.Phone,

//Province = province,

City = ll\_city

},

Sale\_ID = t.Mall\_Trade\_ID,

Post\_Fee = (double)t.Post\_Fee,

Status = t.Status,

Shop = new BShop

{

ID = shop.Shop\_ID,

Title = shop.Name

},

SyncUser = new BUser

{

ID = user.User\_ID,

Name = user.Name,

Mall\_ID = user.Mall\_ID,

Mall\_Name = user.Mall\_Name

},

HasRefound = (bool)t.HasRefound

};//).OrderByDescending(s => s.SaleDateTime).Skip((page - 1) \* pageSize).Take(pageSize).ToList<BSale>();

totalRecords = sObjs.Count();

sales = sObjs.OrderByDescending(s => s.SaleDateTime).ThenByDescending(s => s.Synced).Skip((page - 1) \* pageSize).Take(pageSize).ToList<BSale>(); string[] bsale\_ids = (from sale in sales select sale.Sale\_ID).ToArray<string>();

List<Sale\_Detail> sale\_details = (from sdetail in db.Sale\_Detail where bsale\_ids.Contains(sdetail.Mall\_Trade\_ID) select sdetail).ToList<Sale\_Detail>();

int[] product\_ids = (from sd in sale\_details select sd.Parent\_Product\_ID).ToArray<int>();

string[] mallProduct\_ids = (from sd in sale\_details select sd.Mall\_PID).ToArray<string>();

int[] cproduct\_ids = (from sd in sale\_details select sd.Product\_ID).Distinct<int>().ToArray<int>();

List<Product> dbProducts = (from product in db.Product where product\_ids.Contains(product.Product\_ID) select product).ToList<Product>();

List<Mall\_Product> dbMallProducts=(from p in db.Mall\_Product where mallProduct\_ids.Contains(p.Mall\_ID) select p).ToList<Mall\_Product>();

List<BProductProperty> childs = (from prop in db.Product\_Specifications

join ps in db.Product\_Spec on prop.Product\_Spec\_ID equals ps.Product\_Spec\_ID

join psv in db.Product\_Spec\_Value on prop.Product\_Spec\_Value\_ID equals psv.Product\_Spec\_Value\_ID

where cproduct\_ids.Contains(prop.Product\_ID)

select new BProductProperty

{

ProductID = prop.Product\_ID,

PID = prop.Product\_Spec\_ID,

PName = ps.Name,

PVID = prop.Product\_Spec\_Value\_ID,

PValue = psv.Name

}).ToList<BProductProperty>();

List<Mall\_Product\_Sku> skus=(from sku in db.Mall\_Product\_Sku where mallProduct\_ids.Contains(sku.Mall\_ID)

select sku).ToList<Mall\_Product\_Sku>();

foreach (BSale sale in sales)

{

var os = from order in sale\_details

where order.Mall\_Trade\_ID==sale.Sale\_ID

select new BOrder

{

Mall\_SkuID=order.Mall\_SkuID,

Amount = (double)order.Amount,

Order\_ID = order.Mall\_Order\_ID,

Quantity = order.Quantity,

Price = order.Price,

Parent\_Product\_ID = order.Parent\_Product\_ID,

Product\_ID = order.Product\_ID,

ImageUrl=order.ImageUrl,

Mall\_PID=order.Mall\_PID,

Message=order.SyncResultMessage,

Refound=order.Refound,

Status1=order.Status1

};

sale.Orders = os.ToList<BOrder>();

foreach (BOrder order in sale.Orders)

{ order.Product = (from product in dbProducts

where product.Product\_ID == order.Parent\_Product\_ID

select new BProduct

{

ID = product.Product\_ID,

Title = product.Name

}).FirstOrDefault<BProduct>();

order.MallProduct = (from p in dbMallProducts

where p.Mall\_ID == order.Mall\_PID

select new BMallProduct

{

ID = p.Mall\_ID,

Title = p.Title

}).FirstOrDefault<BMallProduct>();

if (order.MallProduct != null)

{

order.MallProduct.Skus = (from s in skus where s.SKU\_ID == order.Mall\_SkuID select new BMallSku { Properities=s.Properties,PropertiesName=s.Properties\_name }).ToList<BMallSku>();

} if (order.Product != null)

{

order.Product.Properties = (from p in childs

where p.ProductID == order.Product\_ID

select p).ToList<BProductProperty>();

}

}

}

}

return sales;

}

/// <summary>

/// Search back sale details information

/// </summary>

/// <param name="sale\_ids">Trade id list</param>

/// <param name="user\_ids">User id list</param>

/// <param name="stime">Trade start time</param>

/// <param name="etime">Trade endt time</param>

/// <param name="pageIndex">Page index</param>

/// <param name="pageSize">Page size</param>

/// <param name="totalRecords">Total records</param>

/// <returns>List of BBackSaleDetail</returns>

public List<BBackSaleDetail> SearchBackSaleDetails(string[] sale\_ids, int[] user\_ids, int? status, long stime, long etime, int pageIndex, int pageSize, out int totalRecords)

{

if (pageIndex <= 0)

{

pageIndex = 1;

}

if (pageSize <= 0)

{

pageSize = 30;

}

List<BBackSaleDetail> backSaleDetails = new List<BBackSaleDetail>();

List<BBackSale> backSales = new List<BBackSale>();

totalRecords = 0;

using (KuanMaiEntities db = new KuanMaiEntities())

{

int[] cspids = (from c in this.DBChildShops select c.Shop\_ID).ToArray<int>();

var dbBackSales = from sale in db.Back\_Sale

if (cspids != null && cspids.Length > 0)

{

dbBackSales = dbBackSales.Where(bs => (bs.Shop\_ID == this.Shop.Shop\_ID || cspids.Contains(bs.Shop\_ID)));

}

else

{

dbBackSales = dbBackSales.Where(bs => (bs.Shop\_ID == this.Shop.Shop\_ID));

} if (sale\_ids != null && sale\_ids.Length > 0)

{

dbBackSales = dbBackSales.Where(s => sale\_ids.Contains(s.Sale\_ID));

} if (user\_ids != null && user\_ids.Length > 0)

{

dbBackSales = dbBackSales.Where(s => user\_ids.Contains(s.User\_ID));

} if (stime > 0)

{

dbBackSales = dbBackSales.Where(s => s.Back\_Date >= stime);

} if (etime > 0)

{

dbBackSales = dbBackSales.Where(s => s.Back\_Date <= etime);

}

int[] backSaleIds = (from bs in dbBackSales select bs.Back\_Sale\_ID).ToArray<int>();

var tmp = from bsd in db.Back\_Sale\_Detail

join bs in db.Back\_Sale on bsd.Back\_Sale\_ID equals bs.Back\_Sale\_ID into lbs

from l\_bs in lbs.DefaultIfEmpty()

join sale in db.Sale on l\_bs.Sale\_ID equals sale.Mall\_Trade\_ID into lsale

from l\_sale in lsale.DefaultIfEmpty()

join customer in db.Customer on l\_sale.Buyer\_ID equals customer.Customer\_ID into lcustomer

from l\_customer in lcustomer.DefaultIfEmpty()

join product in db.Product on bsd.Parent\_Product\_ID equals product.Product\_ID into lproduct

from l\_product in lproduct.DefaultIfEmpty()

join shop in db.Shop on l\_bs.Shop\_ID equals shop.Shop\_ID into lshop

from l\_shop in lshop.DefaultIfEmpty()

join user in db.User on l\_bs.User\_ID equals user.User\_ID into luser

from l\_user in luser.DefaultIfEmpty()

where backSaleIds.Contains(bsd.Back\_Sale\_ID)

select new BBackSaleDetail

{

Order\_ID=bsd.Order\_ID,

Amount = bsd.Refound,

Created = (int)bsd.Created,

Description = bsd.Description,

BackSaleID = bsd.Back\_Sale\_ID,

ParentProductID = bsd.Parent\_Product\_ID,

Price = bsd.Price,

ProductID = bsd.Product\_ID,

Quantity = bsd.Quantity,

Status = bsd.Status,

BackSale = new BBackSale

{

Amount = l\_bs.Amount,

BackTime = l\_bs.Back\_Date,

Created = l\_bs.Created,

Description = l\_bs.Description,

ID = l\_bs.Back\_Sale\_ID,

Sale = new BSale

{

Sale\_ID = l\_bs.Sale\_ID,

Buyer = new BCustomer

{

ID = l\_customer.Customer\_ID,

Mall\_ID = l\_customer.Mall\_ID,

Mall\_Name = l\_customer.Mall\_Name

},

Amount = l\_sale.Amount

},

Shop = new BShop

{

Title = l\_shop.Name,

ID = l\_shop.Shop\_ID

},

Created\_By = new BUser

{

ID = l\_user.User\_ID,

Mall\_ID = l\_user.Mall\_ID,

Mall\_Name = l\_user.Mall\_Name

}

},

Product=new BProduct

{

Title=l\_product.Name

}

};

if (status != null)

{

tmp = tmp.Where(t=>t.Status==status);

}

tmp = tmp.OrderBy(b=>b.Status).ThenBy(b=>b.BackSaleID);

totalRecords = tmp.Count();

if (totalRecords > 0)

{

backSaleDetails = tmp.Skip((pageIndex-1)\*pageSize).Take(pageSize).ToList<BBackSaleDetail>();

}

}

return backSaleDetails;

}

/// <summary>

/// Search back trades

/// </summary>

/// <param name="sale\_ids">Trade id list</param>

/// <param name="user\_ids">User id list</param>

/// <param name="stime">Trade start time</param>

/// <param name="etime">Trade end time</param>

/// <param name="pageIndex">Page index</param>

/// <param name="pageSize">Page size</param>

/// <param name="totalRecords">Total records</param>

/// <returns>List of BBackSale</returns>

public List<BBackSale> SearchBackSales(string[] sale\_ids, int[] user\_ids, long stime, long etime, int pageIndex, int pageSize, out int totalRecords)

{

if (pageIndex <= 0)

{

pageIndex = 1;

} if (pageSize <= 0)

{

pageSize = 30;

}

List<BBackSale> backSales = new List<BBackSale>();

totalRecords = 0;

using (KuanMaiEntities db = new KuanMaiEntities())

{

int[] cspids = (from c in this.DBChildShops select c.Shop\_ID).ToArray<int>();

if (cspids == null)

{

cspids = new int[] { 0 };

}

var dbBackSales = from sale in db.Back\_Sale

where sale.Shop\_ID == this.Shop.Shop\_ID || sale.Shop\_ID == this.Main\_Shop.Shop\_ID || cspids.Contains(sale.Shop\_ID)

select sale;

if (sale\_ids != null && sale\_ids.Length > 0)

{

dbBackSales = dbBackSales.Where(s => sale\_ids.Contains(s.Sale\_ID));

} if (user\_ids != null && user\_ids.Length > 0)

{

dbBackSales = dbBackSales.Where(s => user\_ids.Contains(s.User\_ID));

} if (stime > 0)

{

dbBackSales = dbBackSales.Where(s => s.Back\_Date >= stime);

} if (etime > 0)

{

dbBackSales = dbBackSales.Where(s => s.Back\_Date <= etime);

} var tmp = from sale in dbBackSales

orderby sale.Shop\_ID ascending

join user in db.User on sale.User\_ID equals user.User\_ID

join mtype in db.Mall\_Type on user.Mall\_Type equals mtype.Mall\_Type\_ID

join order in db.Sale on sale.Sale\_ID equals order.Mall\_Trade\_ID

join customer in db.Customer on order.Buyer\_ID equals customer.Customer\_ID

join shop in db.Shop on sale.Shop\_ID equals shop.Shop\_ID select new BBackSale

{

BackTime = (int)sale.Back\_Date,

Created = sale.Created,

Created\_By = new BUser

{

ID = user.User\_ID,

Mall\_ID = user.Mall\_ID,

Mall\_Name = user.Mall\_Name,

Created = (int)user.Created,

Type = new BMallType { ID = mtype.Mall\_Type\_ID, Name = mtype.Name, Description = mtype.Description }

},

Description = sale.Description,

ID = sale.Back\_Sale\_ID,

Amount = sale.Amount,

Sale = new BSale

{

Amount = order.Amount,

Buyer = new BCustomer

{

ID = customer.Customer\_ID,

Mall\_ID = customer.Mall\_ID,

Mall\_Name = customer.Mall\_Name,

Phone = customer.Phone,

Type = new BMallType { ID=mtype.Mall\_Type\_ID,Name=mtype.Name,Description=mtype.Description},

Address = customer.Address

},

Created = (int)order.Created,

Sale\_ID = order.Mall\_Trade\_ID,

Modified = (int)order.Modified,

Post\_Fee = (double)order.Post\_Fee,

Status = order.Status,

Synced = (int)order.Synced }, Shop = new BShop

{

ID = shop.Shop\_ID,

Title = shop.Name,

Mall\_ID = shop.Mall\_Shop\_ID,

Type = mtype

}

};

totalRecords = tmp.Count();

backSales = tmp.OrderBy(a => a.ID).Skip((pageIndex - 1) \* pageSize).Take(pageSize).ToList<BBackSale>();

int[] bsaleids = (from bsale in backSales select bsale.ID).ToArray<int>();

List<BBackSaleDetail> details = (from detail in db.Back\_Sale\_Detail

where bsaleids.Contains(detail.Back\_Sale\_ID)

select

new BBackSaleDetail{}).ToList<BBackSaleDetail>();

}

return backSales;

}

public List<BSaleSyncLog> SearchTradeSyncLog(int user\_id,int startTime,int endTime,int? syncType,int page,int pageSize,out int totalRecords)

{

List<BSaleSyncLog> log = new List<BSaleSyncLog>();

if(page<=0)

{

page=1;

}

if(pageSize<=0)

{

pageSize=20;

} using (KuanMaiEntities db = new KuanMaiEntities())

{

totalRecords = 0;

int[] child\_shop = (from c in this.DBChildShops select c.Shop\_ID).ToArray<int>();

var tmp = from sync in db.Sale\_SyncTime

select sync;

if (this.Shop.Shop\_ID == this.Main\_Shop.Shop\_ID)

{

tmp = tmp.Where(l => (l.ShopID == this.Shop.Shop\_ID || child\_shop.Contains(l.ShopID)));

}

else

{

tmp = tmp.Where(l => (l.ShopID == this.Shop.Shop\_ID));

} if (user\_id > 0)

{

tmp = tmp.Where(t => t.SyncUser == user\_id);

} if (startTime > 0)

{

tmp = tmp.Where(t => t.LastSyncTime >= startTime);

} if (endTime > 0)

{

tmp = tmp.Where(t => t.LastSyncTime <= endTime);

}

if (syncType != null)

{

tmp = tmp.Where(t => t.SyncType == syncType);

} var tmpSyncLog = from sync in tmp

join user in db.User on sync.SyncUser equals user.User\_ID into lUser

from l\_user in lUser.DefaultIfEmpty()

join shop in db.Shop on sync.ShopID equals shop.Shop\_ID into lShop

from l\_shop in lShop.DefaultIfEmpty()

select new BSaleSyncLog

{

Type=sync.SyncType,

LastModifiedEndTime = sync.LastTradeModifiedEndTime,

LastStartEndTime = sync.LastTradeStartEndTime,

LastSyncTime = sync.LastSyncTime,

Shop = new BShop

{

ID=l\_shop.Shop\_ID,

Title=l\_shop.Name

},

User = new BUser

{

ID=l\_user.User\_ID,

Mall\_Name=l\_user.Mall\_Name,

Mall\_ID=l\_user.Mall\_ID

}

};

totalRecords = tmpSyncLog.Count();

log = tmpSyncLog.OrderByDescending(a => a.LastSyncTime).Skip((page - 1) \* pageSize).Take(pageSize).ToList<BSaleSyncLog>();

}

return log;

}

public void CreateBackSale(BBackSale backSale)

{

if (backSale == null)

{

return;

}

List<BBackSaleDetail> details = backSale.Details;

if (this.CurrentUserPermission.ADD\_BACK\_STOCK == 0)

{

throw new KMJXCException("没有权限进行退货操作");

}

if (backSale.Sale == null || string.IsNullOrEmpty(backSale.Sale.Sale\_ID))

{

throw new KMJXCException("请选择一个销售单进行退货");

} if (details == null)

{

throw new KMJXCException("");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

int[] cspids = (from c in this.DBChildShops select c.Shop\_ID).ToArray<int>();

List<Product> products = (from p in db.Product where p.Shop\_ID == this.Shop.Shop\_ID || p.Shop\_ID == this.Main\_Shop.Shop\_ID || cspids.Contains(p.Shop\_ID) select p).ToList<Product>();

Back\_Sale dbbackSale = new Back\_Sale();

dbbackSale.Back\_Date = backSale.BackTime;

dbbackSale.Back\_Sale\_ID = 0;

dbbackSale.Description = backSale.Description;

dbbackSale.Sale\_ID = backSale.Sale.Sale\_ID;

dbbackSale.Shop\_ID = this.Shop.Shop\_ID;

dbbackSale.User\_ID = this.CurrentUser.ID;

db.Back\_Sale.Add(dbbackSale);

if (dbbackSale.Back\_Sale\_ID <= 0)

{

throw new KMJXCException("退货单创建失败");

}

List<BBackStockDetail> bdetails = new List<BBackStockDetail>();

foreach (BBackSaleDetail sDetail in details)

{

Back\_Sale\_Detail dbSaleDetail = new Back\_Sale\_Detail();

dbSaleDetail.Back\_Sale\_ID = dbbackSale.Back\_Sale\_ID;

dbSaleDetail.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

dbSaleDetail.Description = sDetail.Description;

dbSaleDetail.Parent\_Product\_ID = sDetail.ParentProductID;

if (dbSaleDetail.Parent\_Product\_ID == 0)

{

dbSaleDetail.Parent\_Product\_ID = (from p in products where p.Product\_ID == sDetail.ProductID select p.Parent\_ID).FirstOrDefault<int>();

}

dbSaleDetail.Price = sDetail.Price;

dbSaleDetail.Product\_ID = sDetail.ProductID;

dbSaleDetail.Quantity = sDetail.Quantity;

dbSaleDetail.Status = 0;

db.Back\_Sale\_Detail.Add(dbSaleDetail);

}

db.SaveChanges();

}

}

public void HandleBackSaleDetail\_BackStock(int backSaleID,List<BOrder> borders, int status)

{

if (this.CurrentUserPermission.HANDLE\_BACK\_SALE == 0)

{

throw new KMJXCException("没有处理退货单的权限");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

if (backSaleID <= 0 || borders == null || borders.Count <= 0)

{

throw new KMJXCException("没有找到退货单信息");

}

stockManager.CreateBackStock(backSaleID, borders, status);

}

}

public void HandleBackSaleDetail\_PartialWaste(int backSaleID, List<BOrder> orders, int status)

{

if (this.CurrentUserPermission.HANDLE\_BACK\_SALE == 0)

{

throw new KMJXCException("没有处理退货单的权限");

}

if (backSaleID == 0 || orders == null || orders.Count <= 0)

{

throw new KMJXCException("退货单信息有错误");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

string[] order\_id=(from o in orders select o.Order\_ID).ToArray<string>();

List<Back\_Sale\_Detail> details = (from bsd in db.Back\_Sale\_Detail

where bsd.Back\_Sale\_ID == backSaleID && order\_id.Contains(bsd.Order\_ID)

select bsd).ToList<Back\_Sale\_Detail>(); this.HandleWastageBackSale\_TotalWaste(backSaleID,orders,status);

List<BOrder> bkOrders = new List<BOrder>();

foreach (BOrder o in orders)

{

Back\_Sale\_Detail detail=(from d in details where d.Order\_ID==o.Order\_ID select d).FirstOrDefault<Back\_Sale\_Detail>();

if (detail.Quantity - o.Quantity > 0)

{

bkOrders.Add(new BOrder() { Order\_ID = o.Order\_ID, Quantity = detail.Quantity - o.Quantity });

}

}

this.HandleBackSaleDetail\_BackStock(backSaleID, bkOrders, status);

}

}

public void HandleWastageBackSale\_TotalWaste(int backSaleId,List<BOrder> borders,int status)

{

if (this.CurrentUserPermission.HANDLE\_BACK\_SALE == 0)

{

throw new KMJXCException("没有处理退货单的权限");

}

string[] order\_id = null;

if (backSaleId <= 0 || borders==null || borders.Count==0)

{

throw new KMJXCException("没有退货单信息");

}

order\_id=(from o in borders select o.Order\_ID).ToArray<string>();

using (KuanMaiEntities db = new KuanMaiEntities())

{

Back\_Sale backSake=(from bs in db.Back\_Sale where bs.Back\_Sale\_ID==backSaleId select bs).FirstOrDefault<Back\_Sale>();

if (backSake == null)

{

throw new KMJXCException("编号为:"+backSaleId+" 的退货单信息不存在");

}

List<Back\_Sale\_Detail> details=(from bsd in db.Back\_Sale\_Detail where bsd.Back\_Sale\_ID==backSaleId && order\_id.Contains(bsd.Order\_ID) select bsd).ToList<Back\_Sale\_Detail>();

int[] child\_shop=(from child in this.DBChildShops select child.Shop\_ID).ToArray<int>();

List<Stock\_Waste> wastage=(from waste in db.Stock\_Waste

where waste.Shop\_ID==this.Shop.Shop\_ID || waste.Shop\_ID==this.Main\_Shop.Shop\_ID || child\_shop.Contains(waste.Shop\_ID)

select waste).ToList<Stock\_Waste>();

Leave\_Stock leave\_Stock = (from ls in db.Leave\_Stock where ls.Sale\_ID == backSake.Sale\_ID select ls).FirstOrDefault<Leave\_Stock>(); //no leave no need to set to wastage

if (leave\_Stock == null)

{

return;

}

List<Leave\_Stock\_Detail> leaveDetails = (from ld in db.Leave\_Stock\_Detail where ld.Leave\_Stock\_ID == leave\_Stock.Leave\_Stock\_ID select ld).ToList<Leave\_Stock\_Detail>();

foreach (Back\_Sale\_Detail detail in details)

{

Leave\_Stock\_Detail leaveDetail=(from ld in leaveDetails where ld.Order\_ID==detail.Order\_ID && ld.Product\_ID==detail.Product\_ID select ld).FirstOrDefault<Leave\_Stock\_Detail>();

if (leaveDetail == null)

{

continue;

}

int q=(from o in borders where o.Order\_ID == detail.Order\_ID select o.Quantity).FirstOrDefault<int>();

Stock\_Waste w = (from waste in wastage where waste.Product\_ID == detail.Product\_ID select waste).FirstOrDefault<Stock\_Waste>();

if (w == null)

{

w = new Stock\_Waste();

w.Parent\_ProductID = detail.Parent\_Product\_ID;

w.Price = detail.Price;

w.Product\_ID = w.Product\_ID;

if (w.Parent\_ProductID == 0)

{

w.Parent\_ProductID = w.Product\_ID;

}

w.Quantity = q;

w.Shop\_ID = this.Shop.Shop\_ID;

db.Stock\_Waste.Add(w);

}

else

{

w.Quantity += q;

}

detail.Status = status;

leaveDetail.Status = status;

Sale\_Detail saleDetail=(from sd in db.Sale\_Detail where sd.Mall\_Order\_ID==detail.Order\_ID select sd).FirstOrDefault<Sale\_Detail>();

if (saleDetail != null)

{

saleDetail.Status1 = (int)SaleDetailStatus.REFOUND\_HANDLED;

saleDetail.SyncResultMessage = "退货已经处理";

}

}

db.SaveChanges();

}

}

public List<BBackSaleDetail> GetBackDetails(int backSaleId)

{

List<BBackSaleDetail> details = new List<BBackSaleDetail>();

using (KuanMaiEntities db = new KuanMaiEntities())

{

details = (from bsd in db.Back\_Sale\_Detail

where bsd.Back\_Sale\_ID == backSaleId

select new BBackSaleDetail

{

Created = (int)bsd.Created,

Description = bsd.Description,

Price = double.Parse(bsd.Price.ToString("0.00")),

Quantity = bsd.Quantity,

Product = (from p in db.Product

where bsd.Product\_ID == p.Product\_ID

select new BProduct

{

ID = p.Product\_ID,

Title = p.Name,

Description = p.Description,

Category = new BCategory() { ID = p.Product\_Class\_ID },

Code = p.Code,

CreateTime = p.Create\_Time,

Price = p.Price,

Quantity = (from sp in db.Stock\_Pile where sp.Product\_ID == p.Product\_ID select sp.Quantity).FirstOrDefault<int>()

}).FirstOrDefault<BProduct>()

}).ToList<BBackSaleDetail>();

} return details;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using KM.JXC.DBA;

using KM.JXC.Common.KMException;

using KM.JXC.BL.Open.Interface;

using KM.JXC.BL.Open.TaoBao;

using KM.JXC.BL.Models;

using KM.JXC.Common.Util;

namespace KM.JXC.BL

{

public class BuyManager:BBaseManager

{

public BuyManager(BUser user, int shop\_id, Permission permission)

: base(user, shop\_id,permission)

{

}

public BuyManager(BUser user, Shop shop, Permission permission)

: base(user, shop, permission)

{

}

/// <summary>

/// Search buy orders

/// </summary>

/// <param name="shop\_id"></param>

/// <param name="user\_ids"></param>

/// <param name="startTime"></param>

/// <param name="endTime"></param>

/// <param name="pageIndex"></param>

/// <param name="pageSize"></param>

/// <param name="totalRecords"></param>

/// <returns></returns>

public List<BBuyOrder> SearchBuyOrders(int[] order\_ids, int[] user\_ids, int[] supplier\_ids,int[] product\_ids,string keyword, int startTime, int endTime, int pageIndex, int pageSize, out int totalRecords,bool paging=true)

{

List<BBuyOrder> buyOrders = new List<BBuyOrder>();

totalRecords = 0;

if (pageIndex <= 0)

{

pageIndex = 1;

}

if (pageSize <= 0)

{

pageSize = 30;

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

var bo = from buyOrder in db.Buy\_Order select buyOrder ;

int[] spids=(from sp in this.DBChildShops select sp.Shop\_ID).ToArray<int>();

if (spids != null && spids.Length > 0)

{

bo = bo.Where(a => a.Shop\_ID == this.Shop.Shop\_ID || a.Shop\_ID == this.Main\_Shop.Shop\_ID || spids.Contains(a.Shop\_ID));

}

else

{

bo = bo.Where(a => a.Shop\_ID == this.Shop.Shop\_ID);

}

if (user\_ids !=null && user\_ids.Length>0)

{

bo = bo.Where(a=>user\_ids.Contains(a.User\_ID));

}

if (supplier\_ids != null && supplier\_ids.Length > 0)

{

bo = bo.Where(a => supplier\_ids.Contains(a.Supplier\_ID));

}

if (!string.IsNullOrEmpty(keyword))

{

List<int> ids=(from p in db.Product where p.Name.Contains(keyword) select p.Product\_ID).ToList<int>();

if (product\_ids == null || product\_ids.Length<=0)

{

product\_ids = ids.ToArray<int>();

}

else

{

int[] newIds = ids.ToArray<int>();

int len = product\_ids.Length + newIds.Length;

int[] tmp=new int[len];

product\_ids.CopyTo(tmp, 0);

newIds.CopyTo(tmp,product\_ids.Length);

product\_ids = newIds;

}

}

if (product\_ids != null && product\_ids.Length > 0)

{

int[] buyOrderIds=(from buyo in db.Buy\_Order\_Detail where product\_ids.Contains(buyo.Product\_ID) select buyo.Buy\_Order\_ID).Distinct<int>().ToArray<int>();

if (buyOrderIds != null && buyOrderIds.Length > 0)

{

bo = bo.Where(a => buyOrderIds.Contains(a.Buy\_Order\_ID));

}

}

if (order\_ids != null && order\_ids.Length > 0)

{

bo = bo.Where(a => order\_ids.Contains(a.Buy\_Order\_ID));

}

if (startTime > 0)

{

bo = bo.Where(a=>a.Create\_Date>=startTime);

}

if (endTime > 0)

{

bo = bo.Where(a => a.Create\_Date <= endTime);

}

totalRecords = bo.Count();

if (totalRecords > 0)

{

var efOrders = from border in bo

select new BBuyOrder

{

ID = border.Buy\_Order\_ID,

Created = border.Create\_Date,

Description = border.Description,

EndTime = (int)border.End\_Date,

InsureTime = (int)border.Insure\_Date,

WriteTime = (int)border.Write\_Date,

OrderUser = (from u in db.User

where u.User\_ID == border.Order\_User\_ID

select new BUser

{

ID = u.User\_ID,

Mall\_ID = u.Mall\_ID,

Mall\_Name = u.Mall\_Name,

Name = u.Name

}).FirstOrDefault<BUser>(),

Status = (int)border.Status,

Supplier = (from sp in db.Supplier where sp.Supplier\_ID == border.Supplier\_ID select sp).FirstOrDefault<Supplier>(),

Created\_By = (from u in db.User

where u.User\_ID == border.User\_ID

select new BUser

{

ID = u.User\_ID,

Mall\_ID = u.Mall\_ID,

Mall\_Name = u.Mall\_Name,

Name = u.Name

}).FirstOrDefault<BUser>(),

Shop = (from sp in db.Shop

where sp.Shop\_ID == border.Shop\_ID

select new BShop

{

ID = sp.Shop\_ID,

Mall\_ID = sp.Mall\_Shop\_ID,

Title = sp.Name

}).FirstOrDefault<BShop>()

};

if (paging)

{

efOrders = efOrders.OrderBy(a => a.Status).Skip((pageIndex - 1) \* pageSize).Take(pageSize);

}

buyOrders = efOrders.ToList<BBuyOrder>();

int[] eorder\_ids=(from o in buyOrders select o.ID).ToArray<int>();

int[] eproduct\_ids = (from detail in db.Buy\_Order\_Detail where eorder\_ids.Contains(detail.Buy\_Order\_ID) select detail.Product\_ID).ToArray<int>();

List<BProductProperty> properties = (from pv in db.Product\_Specifications

join prop in db.Product\_Spec on pv.Product\_Spec\_ID equals prop.Product\_Spec\_ID into LProp

from l\_prop in LProp.DefaultIfEmpty()

join propV in db.Product\_Spec\_Value on pv.Product\_Spec\_Value\_ID equals propV.Product\_Spec\_Value\_ID into LPropv

from l\_propv in LPropv.DefaultIfEmpty()

where eproduct\_ids.Contains(pv.Product\_ID)

select new BProductProperty

{

PID = pv.Product\_Spec\_ID,

PName = l\_prop.Name,

ProductID = pv.Product\_ID,

PValue = l\_propv.Name,

PVID = pv.Product\_Spec\_Value\_ID

}).ToList<BProductProperty>();

foreach (BBuyOrder o in buyOrders)

{

var ds = from od in db.Buy\_Order\_Detail

where od.Buy\_Order\_ID == o.ID

select new BBuyOrderDetail

{

Price = od.Price,

Quantity = od.Quantity,

Status = (int)od.Status,

Product = (from pdt in db.Product

where pdt.Product\_ID == od.Product\_ID

select new BProduct

{

ID = pdt.Product\_ID,

Title = pdt.Name,

}).FirstOrDefault<BProduct>(),

Parent\_Product\_ID = od.Parent\_Product\_ID

};

List<BBuyOrderDetail> dss = ds.ToList<BBuyOrderDetail>();

foreach (BBuyOrderDetail d in dss)

{

if (d.Parent\_Product\_ID != d.Product.ID && d.Parent\_Product\_ID > 0)

{

d.Product.Properties = (from prop in properties where prop.ProductID == d.Product.ID select prop).ToList<BProductProperty>();

}

}

o.Details = dss;

if (o.Shop.ID == this.Main\_Shop.Shop\_ID)

{

o.FromMainShop = true;

}

else if (spids != null && spids.Contains(o.Shop.ID))

{

o.FromChildShop = true;

}

}

}

}

return buyOrders;

}

public BBuyOrder GetBuyOrderFullInfo(int order\_id)

{

BBuyOrder order = null;

using (KuanMaiEntities db = new KuanMaiEntities())

{

order = (from o in db.Buy\_Order

join c\_u in db.User on o.User\_ID equals c\_u.User\_ID into LBU

from l\_c\_u in LBU.DefaultIfEmpty()

join o\_u in db.User on o.Order\_User\_ID equals o\_u.User\_ID into LOU

from l\_o\_u in LOU.DefaultIfEmpty()

join suppiler in db.Supplier on o.Supplier\_ID equals suppiler.Supplier\_ID into LSupplier

from l\_supplier in LSupplier.DefaultIfEmpty()

where o.Buy\_Order\_ID == order\_id

select new BBuyOrder

{

Created = o.Create\_Date,

Created\_By = new BUser

{

ID = l\_c\_u.User\_ID,

Name = l\_c\_u.Name,

Mall\_Name = l\_c\_u.Mall\_Name,

Mall\_ID = l\_c\_u.Mall\_ID

},

Description = o.Description,

EndTime = (long)o.End\_Date,

ID = o.Buy\_Order\_ID,

InsureTime = (long)o.Insure\_Date,

OrderUser = new BUser

{

ID = l\_o\_u.User\_ID,

Name = l\_o\_u.Name,

Mall\_Name = l\_o\_u.Mall\_Name,

Mall\_ID = l\_o\_u.Mall\_ID

},

Status = o.Status,

Supplier = l\_supplier,

WriteTime = (long)o.Write\_Date

}).FirstOrDefault<BBuyOrder>();

if (order == null)

{

throw new KMJXCException("编号为:" + order\_id + " 的采购单信息不存在");

}

var tmpODetails = from od in db.Buy\_Order\_Detail

join product in db.Product on od.Product\_ID equals product.Product\_ID into LProduct

from l\_product in LProduct.DefaultIfEmpty()

where od.Buy\_Order\_ID==order\_id

select new BBuyOrderDetail

{

Parent\_Product\_ID = od.Parent\_Product\_ID,

Price = od.Price,

Quantity = od.Quantity,

Product = new BProduct

{

ID = l\_product.Product\_ID,

Title = l\_product.Name

},

Status = od.Status

};

order.Details = tmpODetails.OrderBy(o=>o.Product.ID).ToList<BBuyOrderDetail>();

int[] product\_ids = (from p in order.Details select p.Product.ID).ToArray<int>();

List<BProductProperty> properties = (from pv in db.Product\_Specifications

join prop in db.Product\_Spec on pv.Product\_Spec\_ID equals prop.Product\_Spec\_ID into LProp

from l\_prop in LProp.DefaultIfEmpty()

join propV in db.Product\_Spec\_Value on pv.Product\_Spec\_Value\_ID equals propV.Product\_Spec\_Value\_ID into LPropv

from l\_propv in LPropv.DefaultIfEmpty()

where product\_ids.Contains(pv.Product\_ID)

select new BProductProperty

{

PID = pv.Product\_Spec\_ID,

PName = l\_prop.Name,

ProductID = pv.Product\_ID,

PValue = l\_propv.Name,

PVID = pv.Product\_Spec\_Value\_ID

}).ToList<BProductProperty>();

foreach (BBuyOrderDetail bd in order.Details)

{

if (bd.Product != null)

{

bd.Product.Properties = (from prop in properties where prop.ProductID == bd.Product.ID select prop).ToList<BProductProperty>();

}

}

}

return order;

}

public BBuy GetBuyFullInfo(int buy\_id)

{

BBuy buy = null;

using (KuanMaiEntities db = new KuanMaiEntities())

{

var tmp = from b in db.Buy

join b\_u in db.User on b.User\_ID equals b\_u.User\_ID into LBU

from l\_b\_u in LBU.DefaultIfEmpty()

where b.Buy\_ID==buy\_id

select new BBuy

{

ComeDate = (long)b.Come\_Date,

Created = b.Create\_Date,

ID = b.Buy\_ID,

Description = b.Description,

Status = b.Status,

User = new BUser

{

ID = l\_b\_u.User\_ID,

Name = l\_b\_u.Name != null ? l\_b\_u.Name : l\_b\_u.Mall\_Name,

Mall\_Name = l\_b\_u.Mall\_Name,

Mall\_ID = l\_b\_u.Mall\_ID

},

Order = new BBuyOrder { ID=b.Buy\_Order\_ID}

};

buy = tmp.FirstOrDefault<BBuy>();

if (buy == null)

{

throw new KMJXCException("编号为:"+buy\_id+" 的验货单信息不存在");

}

var tmpBuyDetails = from od in db.Buy\_Detail

join product in db.Product on od.Product\_ID equals product.Product\_ID into LProduct

from l\_product in LProduct.DefaultIfEmpty()

where od.Buy\_ID==buy\_id

select new BBuyDetail

{

Buy\_Order\_ID = od.Buy\_Order\_ID,

CreateDate = od.Create\_Date,

Parent\_Product\_ID = od.Parent\_Product\_ID,

Price = od.Price,

Quantity = od.Quantity,

ProductId = od.Product\_ID,

Product = new BProduct

{

ID = l\_product.Product\_ID,

Title = l\_product.Name

}

};

buy.Details = tmpBuyDetails.ToList<BBuyDetail>();

BBuyOrder order = (from o in db.Buy\_Order

join c\_u in db.User on o.User\_ID equals c\_u.User\_ID into LBU

from l\_c\_u in LBU.DefaultIfEmpty()

join o\_u in db.User on o.Order\_User\_ID equals o\_u.User\_ID into LOU

from l\_o\_u in LOU.DefaultIfEmpty()

join suppiler in db.Supplier on o.Supplier\_ID equals suppiler.Supplier\_ID into LSupplier

from l\_supplier in LSupplier.DefaultIfEmpty()

where o.Buy\_Order\_ID==buy.Order.ID

select new BBuyOrder

{

Created = o.Create\_Date,

Created\_By = new BUser

{

ID = l\_c\_u.User\_ID,

Name = l\_c\_u.Name,

Mall\_Name = l\_c\_u.Mall\_Name,

Mall\_ID = l\_c\_u.Mall\_ID

},

Description = o.Description,

EndTime = (long)o.End\_Date,

ID = o.Buy\_Order\_ID,

InsureTime = (long)o.Insure\_Date,

OrderUser = new BUser

{

ID = l\_o\_u.User\_ID,

Name = l\_o\_u.Name,

Mall\_Name = l\_o\_u.Mall\_Name,

Mall\_ID = l\_o\_u.Mall\_ID

},

Status = o.Status,

Supplier = l\_supplier,

WriteTime = (long)o.Write\_Date

}).FirstOrDefault<BBuyOrder>();

if (order == null)

{

throw new KMJXCException("编号为:" + buy.Order.ID + " 的采购单信息不存在");

}

var tmpODetails = from od in db.Buy\_Order\_Detail

join product in db.Product on od.Product\_ID equals product.Product\_ID into LProduct

from l\_product in LProduct.DefaultIfEmpty()

where od.Buy\_Order\_ID==order.ID

select new BBuyOrderDetail

{

Parent\_Product\_ID = od.Parent\_Product\_ID,

Price = od.Price,

Quantity = od.Quantity,

Product = new BProduct

{

ID = l\_product.Product\_ID,

Title = l\_product.Name

},

Status = od.Status

};

order.Details = tmpODetails.ToList<BBuyOrderDetail>();

int[] product\_ids=(from p in order.Details select p.Product.ID).ToArray<int>();

List<BProductProperty> properties = (from pv in db.Product\_Specifications

join prop in db.Product\_Spec on pv.Product\_Spec\_ID equals prop.Product\_Spec\_ID into LProp

from l\_prop in LProp.DefaultIfEmpty()

join propV in db.Product\_Spec\_Value on pv.Product\_Spec\_Value\_ID equals propV.Product\_Spec\_Value\_ID into LPropv

from l\_propv in LPropv.DefaultIfEmpty()

where product\_ids.Contains(pv.Product\_ID)

select new BProductProperty

{

PID = pv.Product\_Spec\_ID,

PName = l\_prop.Name,

ProductID = pv.Product\_ID,

PValue = l\_propv.Name,

PVID = pv.Product\_Spec\_Value\_ID

}).ToList<BProductProperty>();

foreach (BBuyDetail bd in buy.Details)

{

if (bd.Product != null)

{

bd.Product.Properties=(from prop in properties where prop.ProductID == bd.Product.ID select prop).ToList<BProductProperty>();

}

}

foreach (BBuyOrderDetail bd in order.Details)

{

if (bd.Product != null)

{

bd.Product.Properties = (from prop in properties where prop.ProductID == bd.Product.ID select prop).ToList<BProductProperty>();

}

}

buy.Order = order;

}

return buy;

}

public List<BBuy> SearchBuys(int[] buyids, int[] order\_ids, int[] user\_ids, int[] supplier\_ids, int[] product\_ids, int startTime, int endTime, int pageIndex, int pageSize, out int totalRecords,bool getFullProductInfo=false)

{

List<KM.JXC.BL.Models.BBuy> verifications = new List<Models.BBuy>();

totalRecords = 0;

ProductManager pdtManager=new ProductManager(this.CurrentUser,this.Shop,this.CurrentUserPermission);

if (pageIndex <= 0)

{

pageIndex = 1;

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

var vbo = from vb in db.Buy select vb;

int[] spids = (from sp in this.DBChildShops select sp.Shop\_ID).ToArray<int>();

if (spids != null && spids.Length > 0)

{

vbo = vbo.Where(o => (o.Shop\_ID == this.Shop.Shop\_ID || spids.Contains(o.Shop\_ID)));

}

else

{

vbo = vbo.Where(o => (o.Shop\_ID == this.Shop.Shop\_ID));

}

if (user\_ids != null && user\_ids.Length>0)

{

vbo = vbo.Where(vb1 => user\_ids.Contains(vb1.User\_ID));

}

if (startTime > 0)

{

vbo = vbo.Where(vb1 => vb1.Create\_Date >= startTime);

}

if (order\_ids != null && order\_ids.Length > 0)

{

int[] buyIds=(from bu in db.Buy where order\_ids.Contains(bu.Buy\_Order\_ID) select bu.Buy\_ID).ToArray<int>();

if (buyIds != null && buyIds.Length > 0)

{

vbo = vbo.Where(vb1 => buyIds.Contains(vb1.Buy\_ID));

}

}

if (endTime > 0)

{

vbo = vbo.Where(vb1 => vb1.Create\_Date <= endTime);

}

if (buyids != null && buyids.Length > 0)

{

vbo = vbo.Where(vb1 => buyids.Contains(vb1.Buy\_ID));

}

totalRecords = vbo.Count();

if (totalRecords > 0)

{

var vboo = from vbb in vbo

join border in db.Buy\_Order on vbb.Buy\_Order\_ID equals border.Buy\_Order\_ID into LBorder

from l\_border in LBorder.DefaultIfEmpty()

select new BBuy

{

ID = vbb.Buy\_ID,

Status = (int)vbb.Status,

Order = l\_border != null ? new BBuyOrder { ID = l\_border.Buy\_Order\_ID } : new BBuyOrder { ID = 0 },

ComeDate = (int)vbb.Come\_Date,

Description = vbb.Description,

Created = (int)vbb.Create\_Date,

User = (from u in db.User

where u.User\_ID == vbb.User\_ID

select new BUser

{

ID = u.User\_ID,

Mall\_ID = u.Mall\_ID,

Mall\_Name = u.Mall\_Name,

Parent\_ID = (int)u.Parent\_User\_ID,

Name = u.Name,

Password = u.Password,

}).FirstOrDefault<BUser>(),

Shop = (from sp in db.Shop

where sp.Shop\_ID == vbb.Shop\_ID

select new BShop

{

ID = sp.Shop\_ID,

Title = sp.Name

}).FirstOrDefault<BShop>()

};

vboo = vboo.OrderBy(b => b.ID).Skip((pageIndex - 1) \* pageSize).Take(pageSize);

verifications = vboo.ToList<BBuy>();

int[] orderIds;

orderIds = (from verify in verifications select verify.Order.ID).ToArray<int>();

if (orderIds != null && orderIds.Length > 0)

{

foreach (BBuy b in verifications)

{

b.Details = (from bd in db.Buy\_Detail

where bd.Buy\_ID == b.ID

select new BBuyDetail

{

Buy\_Order\_ID = b.Order.ID,

CreateDate = bd.Create\_Date,

Price = bd.Price,

Quantity = bd.Quantity,

ProductId = bd.Product\_ID,

Product = (from pdt in db.Product

where pdt.Product\_ID == bd.Product\_ID

select new BProduct

{

ID = pdt.Product\_ID,

Title = pdt.Name,

}).FirstOrDefault<BProduct>()

}).ToList<BBuyDetail>();

if (b.Shop.ID == this.Main\_Shop.Shop\_ID)

{

b.FromMainShop = true;

}

else if (spids != null && spids.Contains(b.Shop.ID))

{

b.FromChildShop = true;

}

}

}

}

}

return verifications;

}

public List<BBuyDetail> GetBuyDetails(int buy\_id)

{

List<Models.BBuyDetail> details = new List<BBuyDetail>();

if (buy\_id == 0)

{

throw new KMJXCException("验货单ID必须大于零");

}

BBuy buy = null;

using (KuanMaiEntities db = new KuanMaiEntities())

{

buy = (from b in db.Buy

join border in db.Buy\_Order on b.Buy\_Order\_ID equals border.Buy\_Order\_ID into LBOrder

from l\_border in LBOrder.DefaultIfEmpty()

join shop in db.Shop on b.Shop\_ID equals shop.Shop\_ID into LShop

from l\_shop in LShop.DefaultIfEmpty()

join user in db.User on b.User\_ID equals user.User\_ID into LUser

from l\_user in LUser.DefaultIfEmpty()

where b.Buy\_ID == buy\_id

select new BBuy

{

ComeDate = (long)b.Come\_Date,

Created = b.Create\_Date,

Description = b.Description,

ID = b.Buy\_ID,

Order = new BBuyOrder() { ID = b.Buy\_Order\_ID },

Shop = new BShop

{

ID = l\_shop.Shop\_ID,

Title = l\_shop.Name

},

User = new BUser()

{

ID = l\_user.User\_ID,

Name = l\_user.Name,

Mall\_ID = l\_user.Mall\_ID,

Mall\_Name = l\_user.Mall\_Name

},

Status = b.Status

}).FirstOrDefault<BBuy>();

var tmpDetails = from d in db.Buy\_Detail

join product in db.Product on d.Product\_ID equals product.Product\_ID into LProduct

from l\_product in LProduct.DefaultIfEmpty()

where d.Buy\_ID == buy\_id

select new BBuyDetail

{

Buy\_Order\_ID = d.Buy\_Order\_ID,

CreateDate = d.Create\_Date,

Parent\_Product\_ID = d.Parent\_Product\_ID,

Price = d.Price,

Quantity = d.Quantity,

Product = new BProduct

{

ID = l\_product.Product\_ID,

Title = l\_product.Name

},

ProductId = d.Product\_ID

};

details = tmpDetails.ToList<BBuyDetail>();

int[] product\_ids = (from d in details select d.ProductId).ToArray<int>();

List<BProductProperty> properties = (from pv in db.Product\_Specifications

join prop in db.Product\_Spec on pv.Product\_Spec\_ID equals prop.Product\_Spec\_ID into LProp

from l\_prop in LProp.DefaultIfEmpty()

join propV in db.Product\_Spec\_Value on pv.Product\_Spec\_Value\_ID equals propV.Product\_Spec\_Value\_ID into LPropv

from l\_propv in LPropv.DefaultIfEmpty()

where product\_ids.Contains(pv.Product\_ID)

select new BProductProperty

{

PID = pv.Product\_Spec\_ID,

PName = l\_prop.Name,

ProductID = pv.Product\_ID,

PValue = l\_propv.Name,

PVID = pv.Product\_Spec\_Value\_ID

}).ToList<BProductProperty>();

foreach (BBuyDetail d in details)

{

if (d.Product != null)

{

d.Product.Properties=(from prop in properties where prop.ProductID==d.Product.ID select prop).ToList<BProductProperty>();

}

}

}

return details;

}

public bool CreateNewBuy(BBuy buy)

{

return this.CreateNewBuy(buy,buy.Details);

}

/// <summary>

/// Create a buy which could contains details information

/// </summary>

/// <param name="buy"></param>

/// <param name="details">A list of Buy\_Detail, it could be NULL</param>

/// <returns>TRUE/FALSE</returns>

public bool CreateNewBuy(BBuy buy, List<BBuyDetail> details)

{

bool result = false;

if (this.CurrentUserPermission.ADD\_BUY == 0)

{

throw new KMJXCException("没有权限创建验货单");

}

if (buy.Shop == null || buy.Shop.ID==0)

{

buy.Shop = new BShop() { ID = this.Shop.Shop\_ID };

}

if (buy.User == null || buy.User.ID==0)

{

buy.User = new BUser() { ID = this.CurrentUser.ID };

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

Buy dbBuy = new Buy();

dbBuy.Buy\_Order\_ID = buy.Order.ID;

dbBuy.Come\_Date = buy.ComeDate;

dbBuy.Create\_Date = buy.Created;

dbBuy.Description = buy.Description;

dbBuy.Shop\_ID = buy.Shop.ID;

dbBuy.User\_ID = buy.User.ID;

dbBuy.Status = 0;

db.Buy.Add(dbBuy);

db.SaveChanges();

int buy\_id = (int)dbBuy.Buy\_ID;

if (buy\_id <= 0)

{

throw new KMJXCException("验货单创建失败");

}

if (details != null && details.Count>0)

{

result = true;

result = result & this.CreateNewBuyDetails(dbBuy.Buy\_ID, details);

base.CreateActionLog(new BUserActionLog() { Shop = new BShop { ID = dbBuy.Shop\_ID }, Action = new BUserAction() { Action\_ID = UserLogAction.CREATE\_BUY }, Description = "" });

}

}

return result;

}

public bool VerifyBuyOrder(BBuy buy)

{

bool result = false;

using (KuanMaiEntities db = new KuanMaiEntities())

{

Buy dbBuy=(from dbbuy in db.Buy where dbbuy.Buy\_Order\_ID==buy.Order.ID select dbbuy).FirstOrDefault<Buy>();

if (dbBuy == null)

{

result = this.CreateNewBuy(buy, buy.Details);

}

else

{

result = this.CreateNewBuyDetails(dbBuy.Buy\_ID, buy.Details);

}

//update buy order status

var notVerified = from od in db.Buy\_Order\_Detail where od.Buy\_Order\_ID==buy.Order.ID && od.Status == 0 select od;

if (notVerified.ToList<Buy\_Order\_Detail>().Count == 0)

{

Buy\_Order order=(from o in db.Buy\_Order where o.Buy\_Order\_ID==buy.Order.ID select o).FirstOrDefault<Buy\_Order>();

if (order != null)

{

order.Status = 1;

}

db.SaveChanges();

}

result = true;

}

return result;

}

/// <summary>

/// Create single buy detail

/// </summary>

/// <param name="detail"></param>

/// <returns></returns>

public bool CreateNewBuyDetail(int buyId,BBuyDetail detail)

{

bool ret = false;

if (this.CurrentUserPermission.ADD\_BUY == 0)

{

throw new KMJXCException("没有权限创建验货单信息");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

var d = from bdo in db.Buy\_Detail where bdo.Buy\_Order\_ID == detail.Buy\_Order\_ID && bdo.Product\_ID == detail.Product.ID select bdo;

if (d != null && d.ToList<Buy\_Detail>().Count > 0)

{

throw new KMJXCException("已经验过货");

}

if (buyId<=0)

{

var d2 = from bdo in db.Buy\_Detail where bdo.Buy\_Order\_ID == detail.Buy\_Order\_ID select bdo;

List<Buy\_Detail> existed = d2.ToList<Buy\_Detail>();

if (existed.Count > 0)

{

buyId = existed[0].Buy\_ID;

}

}

Buy\_Detail dbDetail = new Buy\_Detail();

dbDetail.Buy\_ID = buyId;

dbDetail.Buy\_Order\_ID = detail.Buy\_Order\_ID;

dbDetail.Create\_Date = detail.CreateDate;

dbDetail.Price = detail.Price;

dbDetail.Product\_ID = detail.Product.ID;

dbDetail.Quantity = detail.Quantity;

db.Buy\_Detail.Add(dbDetail);

Buy\_Order\_Detail boDetail = null;

var d3 = from bod in db.Buy\_Order\_Detail where bod.Buy\_Order\_ID == detail.Buy\_Order\_ID && bod.Product\_ID == detail.Product.ID select bod;

if (d3.ToList<Buy\_Order\_Detail>().Count == 1)

{

boDetail = d3.ToList<Buy\_Order\_Detail>()[0];

}

boDetail.Status = 1;

db.SaveChanges();

//Verify buy order status

this.VerifyBuyOrder((int)detail.Buy\_Order\_ID);

ret = true;

}

return ret;

}

public bool CreateNewBuyDetails(int buyId,List<BBuyDetail> details)

{

bool result = false;

if (this.CurrentUserPermission.ADD\_BUY == 0)

{

throw new KMJXCException("没有权限创建验货单");

}

if (details == null)

{

throw new KMJXCException("输入错误");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

foreach (BBuyDetail detail in details)

{

var d = from bdo in db.Buy\_Detail where bdo.Buy\_ID==buyId && bdo.Buy\_Order\_ID == detail.Buy\_Order\_ID && bdo.Product\_ID == detail.Product.ID select bdo;

if (d != null && d.ToList<Buy\_Detail>().Count > 0)

{

continue;

}

Buy\_Detail dbDetail = new Buy\_Detail();

dbDetail.Create\_Date = KM.JXC.Common.Util.DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

dbDetail.Buy\_ID = buyId;

dbDetail.Buy\_Order\_ID = detail.Buy\_Order\_ID;

dbDetail.Create\_Date = detail.CreateDate;

dbDetail.Price = detail.Price;

dbDetail.Product\_ID = detail.Product.ID;

dbDetail.Quantity = detail.Quantity;

dbDetail.Parent\_Product\_ID = detail.Parent\_Product\_ID;

db.Buy\_Detail.Add(dbDetail);

Buy\_Order\_Detail boDetail = null;

var d3 = from bod in db.Buy\_Order\_Detail where bod.Buy\_Order\_ID == detail.Buy\_Order\_ID && bod.Product\_ID == detail.Product.ID select bod;

if (d3.ToList<Buy\_Order\_Detail>().Count == 1)

{

boDetail = d3.ToList<Buy\_Order\_Detail>()[0];

}

boDetail.Status = 1;

}

db.SaveChanges();

}

return result;

}

/// <summary>

/// Verify if all buy order details of one buy order have been verified

/// </summary>

/// <param name="buy\_order\_id"></param>

private void VerifyBuyOrder(int buy\_order\_id)

{

using (KuanMaiEntities db = new KuanMaiEntities())

{

int total\_buy\_order\_details = db.Buy\_Order\_Detail.Count(bod=>bod.Buy\_Order\_ID==buy\_order\_id);

int total\_buy\_details = db.Buy\_Detail.Count(bd=>bd.Buy\_Order\_ID==buy\_order\_id);

if (total\_buy\_details == total\_buy\_order\_details)

{

var bo = from bos in db.Buy\_Order where bos.Buy\_Order\_ID == buy\_order\_id select bos;

Buy\_Order buy\_Order = (Buy\_Order)bo;

buy\_Order.Status = 1;

}

db.SaveChanges();

}

}

public bool UpdateBuyOrder(BBuyOrder buy\_Order)

{

bool result = false;

using (KuanMaiEntities db = new KuanMaiEntities())

{

Buy\_Order order = (from o in db.Buy\_Order where o.Buy\_Order\_ID == buy\_Order.ID select o).FirstOrDefault<Buy\_Order>();

if (order == null)

{

throw new KMJXCException("订单不存在");

}

if (this.CurrentUserPermission.UPDATE\_BUY\_ORDER == 0)

{

throw new KMJXCException("没有权限修改采购订单");

}

order.Description = buy\_Order.Description;

order.End\_Date = buy\_Order.EndTime;

order.Insure\_Date = buy\_Order.InsureTime;

if (buy\_Order.OrderUser != null)

{

order.Order\_User\_ID = buy\_Order.OrderUser.ID;

}

else

{

order.Order\_User\_ID = this.CurrentUser.ID;

}

if (buy\_Order.Shop != null)

{

order.Shop\_ID = buy\_Order.Shop.ID;

}

else

{

order.Shop\_ID = this.Shop.Shop\_ID;

}

order.Supplier\_ID = buy\_Order.Supplier.Supplier\_ID;

order.User\_ID = this.CurrentUser.ID;

order.Write\_Date = buy\_Order.WriteTime;

db.SaveChanges();

result = true;

if (result)

{

result = this.DeleteBuyOrderDetails(buy\_Order.ID);

if (result)

{

result=this.CreateNewBuyOrderDetails(buy\_Order.ID, buy\_Order.Details);

}

}

}

return result;

}

public bool CreateNewBuyOrder(BBuyOrder buy\_Order)

{

return this.CreateNewBuyOrder(buy\_Order, buy\_Order.Details);

}

/// <summary>

/// Add new buy order

/// </summary>

/// <param name="buy\_Order">Buy\_Order object</param>

/// <returns>TRUE/FALSE</returns>

public bool CreateNewBuyOrder(BBuyOrder buy\_Order, List<BBuyOrderDetail> details)

{

bool result = false;

if (this.CurrentUserPermission.ADD\_BUY\_ORDER == 0)

{

throw new KMJXCException("没有权限创建采购单合同");

}

if (buy\_Order.Shop == null || buy\_Order.Shop.ID<=0 || buy\_Order.Supplier==null || buy\_Order.Supplier.Supplier\_ID==0)

{

throw new KMJXCException("Buy\_Order 对象属性值不全，缺少店铺ID和供应商ID",ExceptionLevel.SYSTEM);

}

if (buy\_Order.Supplier == null || buy\_Order.Supplier.Supplier\_ID == 0)

{

throw new KMJXCException("供应商不能为空");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

Buy\_Order order = new Buy\_Order();

order.Buy\_Order\_ID = buy\_Order.ID;

order.Create\_Date = buy\_Order.Created;

order.Description = buy\_Order.Description;

order.End\_Date = buy\_Order.EndTime;

order.Insure\_Date = buy\_Order.InsureTime;

if (buy\_Order.OrderUser != null)

{

order.Order\_User\_ID = buy\_Order.OrderUser.ID;

}

else

{

order.Order\_User\_ID = this.CurrentUser.ID;

}

if (buy\_Order.Shop != null)

{

order.Shop\_ID = buy\_Order.Shop.ID;

}

else

{

order.Shop\_ID = this.Shop.Shop\_ID;

}

order.Status = 0;

order.Supplier\_ID = buy\_Order.Supplier.Supplier\_ID;

order.User\_ID = this.CurrentUser.ID;

order.Write\_Date = buy\_Order.WriteTime;

db.Buy\_Order.Add(order);

db.SaveChanges();

if (order.Buy\_Order\_ID > 0)

{

result = true;

if (details != null && details.Count > 0)

{

result = result&this.CreateNewBuyOrderDetails(order.Buy\_Order\_ID, details);

base.CreateActionLog(new BUserActionLog() { Shop = new BShop { ID = order.Shop\_ID }, Action = new BUserAction() { Action\_ID = UserLogAction.CREATE\_BUY\_ORDER }, Description = "" });

}

}

}

return result;

}

/// <summary>

/// Delete buy order

/// </summary>

/// <param name="buy\_order\_id"></param>

/// <returns></returns>

public bool DeleteBuyOrder(int buy\_order\_id)

{

bool result = false;

if (buy\_order\_id == 0)

{

throw new KMJXCException("采购单合同ID不能为空");

}

if (this.CurrentUserPermission.DELETE\_BUY\_ORDER == 0)

{

throw new KMJXCException("没有权限删除采购单合同");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

var od = from odo in db.Buy\_Order\_Detail where odo.Buy\_Order\_ID == buy\_order\_id select odo;

if (od.ToList<Buy\_Order\_Detail>().Count > 0)

{

throw new KMJXCException("采购单合同里包含产品详细订单，所以不能删除");

}

Buy\_Order buy\_Order = new Buy\_Order() { Buy\_Order\_ID = buy\_order\_id };

db.Buy\_Order.Attach(buy\_Order);

db.Buy\_Order.Remove(buy\_Order);

db.SaveChanges();

result = true;

}

return result;

}

public bool CreateNewBuyOrderDetail(int buyOrderId,BBuyOrderDetail detail)

{

if (this.CurrentUserPermission.ADD\_BUY\_ORDER == 0)

{

throw new KMJXCException("没有权限创建采购单合同");

}

bool result = false;

if (detail == null || detail.BuyOrder.ID == 0 || detail.Product.ID == 0 || detail.Quantity == 0)

{

throw new KMJXCException("采购单合同不能为空，产品不能为空，数量不能为零");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

var o = from oo in db.Buy\_Order where oo.Buy\_Order\_ID == detail.BuyOrder.ID select oo;

Buy\_Order order = (Buy\_Order)o;

if (order.Status != 0)

{

throw new KMJXCException("此采购单已经验货或者验货未通过，所以不能继续添加");

}

Buy\_Order\_Detail dbOrderDetail = new Buy\_Order\_Detail();

if (buyOrderId <= 0)

{

dbOrderDetail.Buy\_Order\_ID = detail.BuyOrder.ID;

}

else

{

dbOrderDetail.Buy\_Order\_ID = buyOrderId;

}

dbOrderDetail.Price = detail.Price;

dbOrderDetail.Product\_ID = detail.Product.ID;

dbOrderDetail.Quantity = detail.Quantity;

dbOrderDetail.Status = detail.Status;

db.Buy\_Order\_Detail.Add(dbOrderDetail);

db.SaveChanges();

result = true;

}

return result;

}

public bool CreateNewBuyOrderDetails(int buyOrderId, List<BBuyOrderDetail> details)

{

if (this.CurrentUserPermission.ADD\_BUY\_ORDER == 0)

{

throw new KMJXCException("没有权限创建采购单合同");

}

bool result = false;

if (details != null)

{

using (KuanMaiEntities db = new KuanMaiEntities())

{

if (buyOrderId <= 0)

{

var o = from oo in db.Buy\_Order where oo.Buy\_Order\_ID == details[0].BuyOrder.ID select oo;

Buy\_Order order = (Buy\_Order)o;

if (order.Status != 0)

{

throw new KMJXCException("此采购单已经验货或者验货未通过，不能继续添加采购合同产品信息");

}

buyOrderId = order.Buy\_Order\_ID;

}

foreach (BBuyOrderDetail detail in details)

{

if (detail.Quantity <= 0)

{

continue;

}

Buy\_Order\_Detail dbOrderDetail = new Buy\_Order\_Detail();

dbOrderDetail.Buy\_Order\_ID = buyOrderId;

dbOrderDetail.Price = detail.Price;

dbOrderDetail.Product\_ID = detail.Product.ID;

dbOrderDetail.Quantity = detail.Quantity;

dbOrderDetail.Status = detail.Status;

dbOrderDetail.Parent\_Product\_ID = detail.Parent\_Product\_ID;

db.Buy\_Order\_Detail.Add(dbOrderDetail);

}

db.SaveChanges();

result = true;

}

}

return result;

}

/// <summary>

/// Delete single buy order detail from buy order

/// </summary>

/// <param name="detail">Buy\_Order\_Detail object</param>

/// <returns>TRUE/FALSE</returns>

public bool DeleteBuyOrderDetail(BBuyOrderDetail detail)

{

bool result = false;

if (this.CurrentUserPermission.DELETE\_BUY\_ORDER == 0)

{

throw new KMJXCException("没有权限删除采购单合同里的产品信息");

}

if (detail == null || detail.BuyOrder==null || detail.BuyOrder.ID == 0)

{

throw new KMJXCException("请选择采购单合同");

}

if (detail.Product == null || detail.Product.ID == 0)

{

throw new KMJXCException("请选择产品");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

var o = from oo in db.Buy\_Order where oo.Buy\_Order\_ID == detail.BuyOrder.ID select oo;

Buy\_Order order = (Buy\_Order)o;

if (order.Status == 1)

{

throw new KMJXCException("此采购单已经验货通过，不能删除任何产品采购信息");

}

if (order.Status == 3)

{

throw new KMJXCException("此采购单被冻结，不能删除任何产品采购信息");

}

Buy\_Order\_Detail dbOrderDetail = (from dbdetail in db.Buy\_Order\_Detail

where dbdetail.Buy\_Order\_ID==detail.BuyOrder.ID && dbdetail.Product\_ID==detail.Product.ID

select dbdetail).FirstOrDefault<Buy\_Order\_Detail>();

if (dbOrderDetail != null)

{

db.Buy\_Order\_Detail.Remove(dbOrderDetail);

db.SaveChanges();

}

result = true;

}

return result;

}

public bool DeleteBuyOrderDetails(int buyOrderID)

{

bool result = false;

if (this.CurrentUserPermission.DELETE\_BUY\_ORDER == 0)

{

throw new KMJXCException("没有权限删除采购单产品采购信息");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

Buy\_Order order = (from oo in db.Buy\_Order where oo.Buy\_Order\_ID == buyOrderID select oo).FirstOrDefault<Buy\_Order>();

if (order == null)

{

throw new KMJXCException("采购订单不存在");

}

if (order.Status == 1)

{

throw new KMJXCException("此采购单已经验货通过，不能删除任何产品采购信息");

}

List<Buy\_Order\_Detail> details=(from detail in db.Buy\_Order\_Detail where detail.Buy\_Order\_ID==buyOrderID select detail).ToList<Buy\_Order\_Detail>();

foreach (Buy\_Order\_Detail d in details)

{

db.Buy\_Order\_Detail.Remove(d);

}

db.SaveChanges();

result = true;

}

return result;

}

/// <summary>

/// Delete multiple buy order details from one buy order

/// </summary>

/// <param name="details">Buy\_Order\_Detail list object</param>

/// <returns>TRUE/FALSE</returns>

public bool DeleteBuyOrderDetails(List<BBuyOrderDetail> details)

{

bool result = false;

if (this.CurrentUserPermission.DELETE\_BUY\_ORDER == 0)

{

throw new KMJXCException("没有权限删除采购单产品采购信息");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

var o = from oo in db.Buy\_Order where oo.Buy\_Order\_ID == details[0].BuyOrder.ID select oo;

Buy\_Order order = (Buy\_Order)o;

if (order.Status == 1)

{

throw new KMJXCException("此采购单已经验货通过，不能删除任何产品采购信息");

}

if (order.Status == 3)

{

throw new KMJXCException("此采购单被冻结，不能删除任何产品采购信息");

}

foreach (BBuyOrderDetail detail in details)

{

Buy\_Order\_Detail dbOrderDetail = (from dbdetail in db.Buy\_Order\_Detail

where dbdetail.Buy\_Order\_ID == detail.BuyOrder.ID && dbdetail.Product\_ID == etail.Product.ID

select dbdetail).FirstOrDefault<Buy\_Order\_Detail>();

if (dbOrderDetail != null)

{

db.Buy\_Order\_Detail.Remove(dbOrderDetail);

}

}

db.SaveChanges();

result = true;

}

return result;

}

public bool SaveBuyPriceDetails(List<BBuyPriceDetail> details, int buyPriceId)

{

bool result = false;

if (this.CurrentUserPermission.CREATE\_BUY\_PRICE == 0 || this.CurrentUserPermission.UPDATE\_BUY\_PRICE==0)

{

throw new KMJXCException("没有权限创建或者修改采购询价单内容");

}

if (details == null)

{

throw new KMJXCException("输入错误");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

List<Buy\_Price\_Detail> priceDetails=(from p in db.Buy\_Price\_Detail where p.Buy\_Price\_ID==buyPriceId select p).ToList<Buy\_Price\_Detail>();

foreach (BBuyPriceDetail detail in details)

{

bool isNew = false;

if (detail.Supplier == null || detail.Product==null)

{

continue;

}

Buy\_Price\_Detail dbDetail = (from pd in priceDetails

where pd.Buy\_Price\_ID==buyPriceId && pd.Product\_ID==detail.Product.ID && pd.Supplier\_ID==detail.Supplier.ID select pd).FirstOrDefault<Buy\_Price\_Detail>();

if (dbDetail == null)

{

isNew = true;

dbDetail = new Buy\_Price\_Detail();

}

dbDetail.Price = detail.Price;

if (isNew)

{

dbDetail.Buy\_Price\_ID = buyPriceId;

dbDetail.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

if (detail.Created > 0)

{

dbDetail.Created = detail.Created;

}

dbDetail.Description = detail.Desc;

if (dbDetail.Description == null)

{

dbDetail.Description = "";

}

dbDetail.Parent\_Product\_ID = detail.Product.ParentID;

dbDetail.Product\_ID = detail.Product.ID;

dbDetail.Supplier\_ID = detail.Supplier.ID;

dbDetail.PricedUser\_ID = 0;

db.Buy\_Price\_Detail.Add(dbDetail);

}

}

db.SaveChanges();

result = true;

}

return result;

}

public bool CreateBuyPrice(BBuyPrice buyPrice)

{

bool result = false;

if (this.CurrentUserPermission.CREATE\_BUY\_PRICE == 0)

{

throw new KMJXCException("没有权限创建采购询价单");

}

if (buyPrice == null)

{

throw new KMJXCException("输入不正确");

}

using (KuanMaiEntities db = new KuanMaiEntities())

{

Buy\_Price dbBuyPrice = new Buy\_Price();

if (buyPrice.Created <= 0)

{

dbBuyPrice.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

}

dbBuyPrice.Shop\_ID = this.Shop.Shop\_ID;

if (buyPrice.Shop != null && buyPrice.Shop.ID > 0)

{

dbBuyPrice.Shop\_ID = buyPrice.Shop.ID;

}

dbBuyPrice.User\_ID = this.CurrentUser.ID;

if (buyPrice.User != null && buyPrice.User.ID > 0)

{

dbBuyPrice.User\_ID = buyPrice.User.ID;

}

dbBuyPrice.Created = DateTimeUtil.ConvertDateTimeToInt(DateTime.Now);

dbBuyPrice.Title = buyPrice.Title;

dbBuyPrice.Description = buyPrice.Desc;

db.Buy\_Price.Add(dbBuyPrice);

db.SaveChanges();

result = true;

if (dbBuyPrice.ID > 0 && buyPrice.Details!=null && buyPrice.Details.Count>0)

{

result = result & this.SaveBuyPriceDetails(buyPrice.Details, dbBuyPrice.ID);

}

}

return result;

}

}